(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－76118
(P2000－76118A)
(43)公開日　平成12年３月14日(2000.3.14)

(51)Int.Cl.'	識別記号	ＦＩ		テーマコード'(参考)
Ｇ０６Ｆ　12/00	５４５	Ｇ０６Ｆ　12/00	５４５Ｂ	５Ｂ０８２
			５４５Ｍ	
	５４６		５４６Ｋ	

審査請求　未請求　請求項の数16　ＯＬ　（全 40 頁）

(21)出願番号　特願平10－249235

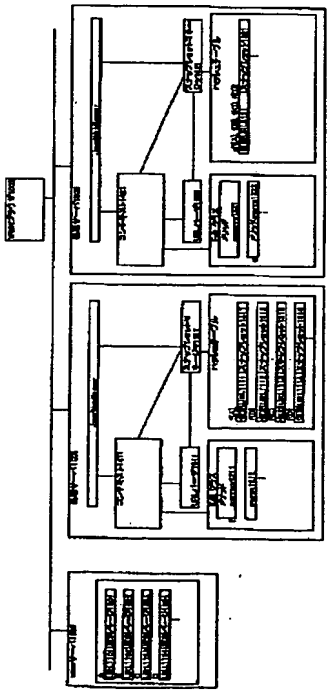
(22)出願日　平成10年９月３日(1998.9.3)

(71)出願人　000005496
富士ゼロックス株式会社
東京都港区赤坂二丁目17番22号
(72)発明者　齋藤　淳
神奈川県足柄上郡中井町境430　グリーンテクなかい　富士ゼロックス株式会社内
(72)発明者　川邉　恵久
神奈川県足柄上郡中井町境430　グリーンテクなかい　富士ゼロックス株式会社内
(74)代理人　100086531
弁理士　澤田　俊夫

Ｆターム(参考)　5B082 HA03 HA05

(54)【発明の名称】　分散ファイル処理装置および分散ファイル処理方法

(57)【要約】
【課題】　複製を有効に利用してファイルサーバへの負荷や、経路上の通信トラフィックを軽減した分散ファイル処理装置および方法を提供する。
【解決手段】　原料を編集加工する手続きを表す手続き名等で修飾して構成した仮想ＵＲＬに基づいて、原料ファイルの処理を行うコンテキスト手段と、コンテキスト手段において処理を行った原料ファイルの処理結果を保持する結果保持手段を有し、さらに、結果保持手段の保持する処理結果の有効性を判定する結果管理手段とを有する。結果管理手段は、処理結果に対応する原料ファイル名、手続き名、手続き処理の処理実行日時、手続き処理実行を要求した要求固有の処理要求識別子等に基づいて処理結果の有効性を判定する。



1

【特許請求の範囲】

【請求項１】　コンピュータをネットワークで接続した分散コンピュータシステムにおける分散ファイル処理装置において、

前記ネットワーク上に接続された装置に保持されたデータファイル原料を識別可能な原料ファイル名を、原料を編集加工する手続きを表す手続き名と、手続きのパラメータと、手続き名を解釈し手続きを動作させるコンピュータおよび動作環境を定めるコンテキスト名とで修飾して構成した修飾ファイル名を解釈して、原料ファイル名と手続き名を取り出し、該取り出した原料ファイル名に対応する原料ファイルのデータを入力し、前記取り出した手続き名に対応する手続きを起動して、原料ファイルを該起動した手続きによって処理を行うコンテキスト手段と、

前記コンテキスト手段において起動した前記手続きによって処理を行った原料ファイルの処理結果を保持する処理結果保持手段と、

前記処理結果保持手段の保持する前記処理結果を取り出して出力する結果出力手段と、

前記処理結果保持手段の保持する処理結果の有効性を判定する処理結果管理手段と、

を有することを特徴とする分散ファイル処理装置。

【請求項２】　前記処理結果管理手段は、

前記処理結果保持手段の保持する処理結果に対応する原料ファイル名およびコンテキスト手段において実行された手続き名に基づいて前記処理結果の有効性を判定する構成を有することを特徴とする請求項１記載の分散ファイル処理装置。

【請求項３】　前記処理結果管理手段は、

前記コンテキスト手段による手続き処理の処理実行日時、

前記コンテキスト手段による手続き処理実行を要求した要求固有の処理要求識別子、の少なくともいずれかに基づいて前記処理結果の有効性を判定する構成を有することを特徴とする請求項２記載の分散ファイル処理装置。

【請求項４】　前記処理結果管理手段は、

前記原料ファイルの更新日時と、前記コンテキスト手段による手続き処理の処理実行日時とを比較し、該比較結果に基づいて前記処理結果の有効性を判定する構成を有することを特徴とする請求項２記載の分散ファイル処理装置。

【請求項５】　前記コンテキスト手段が、前記修飾ファイル名の解釈において、要求処理結果の処理時刻指定子が含まれると判定した場合は、

前記処理結果管理手段は、前記時刻指定子の示す時間範囲に前記処理日時が対応するか否かに基づいて、前記処理日時が対応づけられた処理結果の有効性を判定する構成を有することを特徴とする請求項３記載の分散ファイル処理装置。

2

【請求項６】　前記ネットワークは、

複数のコンテキスト手段、該複数のコンテキスト手段に対応して設けられた複数の処理結果保持手段、および該複数の処理結果保持手段に対応して設けられた複数の処理結果管理手段を接続した構成を有し、

前記複数のコンテキスト手段は、それぞれ独立に手続き処理が可能であり、

ネットワーク上に出力された処理要求に対応する処理結果の有効性を判定する第１の処理結果管理手段は、

該処理要求に指定された原料ファイルおよび手続きに対応する処理結果を保持するネットワーク上の第２の処理結果保持手段を管理する第２の処理結果管理手段に処理結果の有効性についての問い合わせを実行する構成を有することを特徴とする請求項１記載の分散ファイル処理装置。

【請求項７】　前記コンテキスト手段における修飾ファイル名の解釈において、前記処理結果保持手段に保持された処理結果の無効化または破棄いずれかの指示を定めた所定のパラメータが含まれることが解析されたことを条件として、

前記処理結果管理手段は、前記処理結果保持手段に格納された処理結果の無効化または破棄いずれかの処理を実行する構成を有することを特徴とする請求項１記載の分散ファイル処理装置。

【請求項８】　前記処理結果管理手段による前記処理結果保持手段に格納された処理結果の無効化または破棄いずれかの処理の実行に伴い、

前記コンテキスト手段は、前記無効化または破棄を行った処理結果に対応する原料ファイルを取得して、新たに処理を実行して前記処理結果保持手段に保持する構成を有することを特徴とする請求項７記載の分散ファイル処理装置。

【請求項９】　前記コンテキスト手段における原料ファイルに対する前記手続きによる処理は、処理要求タイミングとは独立に非同期的処理として実行するとともに、該手続きによって処理された処理結果には該処理結果に対応する処理結果識別子を生成し、該処理結果識別子を処理結果に対応させて前記処理結果保持手段に保持する構成としたことを特徴とする請求項１に記載の分散ファイル処理装置。

【請求項１０】　前記コンテキスト手段は、前記修飾ファイル名の解釈において、処理の要求が非同期的であることを示す指定データが含まれる判定がなされたことを条件として、原料ファイルに対する前記手続きを処理要求タイミングとは独立に非同期的処理として実行する構成を有することを特徴とする請求項９に記載の分散ファイル処理装置。

【請求項１１】　前記コンテキスト手段が前記修飾ファイル名の解釈において、処理の要求条件に完全一致しないが、部分一致する処理結果が前記処理結果保持手段に

含まれると判定した場合において、
前記結果出力手段は、前記処理結果保持手段に含まれる部分一致する処理結果を出力する構成を有することを特徴とする請求項１に記載の分散ファイル処理装置。
【請求項１２】　コンピュータをネットワークで接続した分散コンピュータシステムにおける分散ファイル処理方法において、
前記ネットワーク上に接続された装置に保持されたデータファイル原料を識別可能な原料ファイル名を、原料を編集加工する手続きを表す手続き名と、手続きのパラメータと、手続き名を解釈し手続きを動作させるコンピュータおよび動作環境を定めるコンテキスト名とで修飾して構成した修飾ファイル名を解釈して、原料ファイル名と手続き名を取り出し、該取り出した原料ファフイル名に対応する原料ファイルのデータを入力し、前記取り出した手続き名に対応する手続きを起動して、原料ファイルを該起動した手続きによって処理を行うコンテキスト手段による手続き処理ステップと、
前記手続き処理ステップにおいて起動した前記手続きによって処理を行った原料ファイルの処理結果を処理結果保持手段に保持する結果保持ステップと、
前記処理結果保持手段の保持する処理結果の有効性を判定する処理結果有効性判定ステップと、
前記処理結果有効性判定ステップにおいて有効と判定された処理結果であり、前記処理結果保持手段に保持された処理結果を取り出して出力する結果出力ステップと、
を有することを特徴とする分散ファイル処理方法。
【請求項１３】　前記処理結果有効性判定ステップは、前記処理結果保持手段の保持する処理結果に対応する前記原料ファイル名および前記コンテキスト手段において実行された手続き名に基づいて前記処理結果の有効性を判定するステップを含むことを特徴とする請求項１２記載の分散ファイル処理方法。
【請求項１４】　前記処理結果有効性判定ステップは、前記コンテキスト手段による手続き処理の処理実行日時、
前記コンテキスト手段による手続き処理の実行を要求した要求固有の処理要求識別子の少なくともいずれかに基づいて前記処理結果の有効性を判定するステップを含むことを特徴とする請求項１３記載の分散ファイル処理方法。
【請求項１５】　前記処理結果有効性判定ステップは、前記原料ファイルの更新日時と、前記コンテキスト手段による手続き処理の処理実行日時を示す処理日時とを比較し、該比較結果に基づいて処理結果の有効性判定を行うステップを含むことを特徴とする請求項１３記載の分散ファイル処理方法。
【請求項１６】　前記コンテキスト手段における原料ファイルに対する手続き処理は、処理要求タイミングとは独立に非同期的処理として実行するとともに、該手続きによって処理された処理結果には該処理結果に対応する処理結果識別子を生成し、該処理結果識別子を処理結果に対応させて前記処理結果保持手段に保持することを特徴とする請求項１２に記載の分散ファイル処理方法。

【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は分散ファイル処理装置および処理方法に関する。さらに詳細には複数のコンピュータが接続されたデータ通信ネットワークを介してネットワーク上のサーバー等に分散して存在するデータに対して効率のよいアクセスを可能とし、迅速な処理データの抽出を可能とした分散ファイル処理装置及び分散ファイル処理方法に関する。
【０００２】
【従来の技術】ネットワークを介する分散ファイルアクセスシステムにおいては昨今、ますますその迅速、正確性に対する要求が高まってきている。従来から、様々な分散ファイル処理システムが提案され、また実行されているが、その代表的なものを以下に説明する。
【０００３】［従来技術１］従来の分散ファイルアクセスシステムにおける一つの一般的方式として、ファイルを提供するファイルサーバとファイルをアクセスするクライアントのネットワークの経路上に、代理サーバや中継サーバ（以降代理サーバ）を分散的に配置して、クライアントはこれを経由してファイルにアクセスする方式がある。例えば、ＦＴＰミラーサーバなどがこれに相当する。
【０００４】このような代理サーバを用いたシステムでは、あらかじめファイルの複製を提供することで、ファイルサーバへの負荷や、ネットワーク経路上の通信トラフィックを軽減できるというメリットがある。
【０００５】また、あらかじめファイルの複製をサーバに用意しておくのではなく、クライアントからアクセスされたファイルに対して動的に複製を作成し、その後のアクセスに提供するスナップショット方式も考案されている。例えば、ＲＦＣ２１８７で述べられているＨＴＴＰ中継サーバプログラムＳｑｕｉｄ（ｈｔｔｐ：／／ｓｑｕｉｄ．ｎｌａｎｒ．ｎｅｔ／Ｓｑｕｉｄ／において配布されている）においては、複製の更新や無効化は、複製の生成時刻、オリジナルのファイルの生成時刻、などのタイムスタンプ情報に基づく計算により必要と判定されたときに行われる。
【０００６】このようなスナップショット方式では、例えば、次のような手順でファイル複製の管理が行われる。クライアントからのファイルアクセス要求が発生し、要求されたファイルの複製が代理サーバにあった場合に、前記タイムスタンプ情報に基づく計算を行い、複製の新しさを判定する。複製が十分新しい場合以外は、要求されたファイルのオリジナルのタイムスタンプと複製のタイムスタンプとを比較して、複製の方が新しい場

合は複製を返す。複製の方が古い場合は、古いと判定された複製を無効化し、新たにオリジナルファイルをコピーして、複製を作成し返す。このような動的に複製を作成する方式では、あらかじめ複製を用意する方式に比べ、複製の管理が自動化できるので管理コストが軽減され、代理サーバの増設も容易である。

【０００７】ところが、WWWのＣＧＩで代表されるような、ファイルサーバ上でアクセス時にオンデマンド型で生成されるファイル（例えば、　今日の天気　をファイル内の情報に含む場合）では、タイムスタンプ情報は、アクセス時に付与される。タイムスタンプ情報のみを用いて複製を管理する方式では、複製とオリジナルの比較を行うための適当なタイムスタンプを決定する機構がないため、複製が有効であるかとうかの判定ができない。そのため、動的に生成されるファイルに関して複製を有効に利用することは困難である。この種の技術を従来技術１とする。

【０００８】［従来技術２］特開平８－２９２９１０には、原料となる資源名から、それを加工する手続きを表す手続き名と、手続きに対するパラメータと、手続き名を解釈し手続きを起動する装置をあらわすコンテキスト識別子とで修飾して構成した資源名を解釈して、原料資源へのアクセスを行い、手続きを起動して、原料資源の処理を行う資源管理装置および資源管理方式が開示されている。

【０００９】この特開平８－２９２９１０に開示された資源管理装置および資源管理方式では、資源名に埋め込まれた手続き名や原料資源名が、分散的に解決され、手続き同士がネットワーク上でパイプライン的に結合される。しかし、手続きが多段に結合されたり、複数の入力を持つ手続きの場合はパイプラインがツリー構造のように構成されることを考えると、ネットワーク上のバースト的な通信エラーでいずれか一個所の通信が失敗した場合、最終的な処理結果である合成資源が得られないという欠点がある。通信に失敗して結果が得られなかった場合は、再実行を行って結果を得ることもできるが、一つの合成資源を得るのに複数のネットワーク通信が必要なので、信頼性の低いネットワークでは、いずれかの通信が高い確率で失敗し、結果を得るまでに何度も再実行を繰り返す必要があったり、合成資源が得られないこともある。この種の技術を従来技術２とする。

【００１０】

【発明が解決しようとする課題】分散ファイル処理において、作成された複製を有効に利用してファイルサーバへの負荷や、ネットワーク経路上の通信トラフィックを軽減することを目標とすることを考慮すると、上述した従来技術の構成では以下に説明する問題点が発生する。

【００１１】まず、従来技術１は、上述のようにWWWのアクセスにおいて、複製を利用して通信トラフィックを減らすなどの効果は提供するが、クライアント等からのリクエストを受けたときにサーバ上で動的に生成される文書に関しては複製を取り扱うのが困難となる。

【００１２】また、従来技術２は分散ファイルアクセスの手順を用いて分散ファイル処理を実現でき、各種の処理を組み合わせたサービスを提供することが可能となるが、処理結果がパイプラインで複数組み合わせることを必要とするため、信頼性の低いネットワークにおいて高い確率で失敗が起きやすく、実用性が低いという問題点を有する。

【００１３】本発明はこのような従来技術の有する各種の問題点を解決するものであり、複製を有効に利用してファイルサーバへの負荷や、経路上の通信トラフィックを軽減することを達成することを目的とする。

【００１４】

【課題を解決するための手段】本発明は上記の目的を達成するものであり、コンピュータをネットワークで接続した分散コンピュータシステムにおける分散ファイル処理装置において、ネットワーク上に接続された装置に保持されたデータファイル原料を識別可能な原料ファイル名を、原料を編集加工する手続きを表す手続き名と、手続きのパラメータと、手続き名を解釈し手続きを動作させるコンピュータおよび動作環境を定めるコンテキスト名とで修飾して構成した修飾ファイル名を解釈して、原料ファイル名と手続き名を取り出し、該取り出した原料ファファイル名に対応する原料ファイルのデータを入力し、取り出した手続き名に対応する手続きを起動して、原料ファイルを該起動した手続きによって処理を行うコンテキスト手段と、コンテキスト手段において起動した手続きによって処理を行った原料ファイルの処理結果を保持する処理結果保持手段と、処理結果保持手段の保持する処理結果を取り出して出力する結果出力手段と、結果保持手段の保持する処理結果の有効性を判定する処理結果管理手段とを有することを特徴とする。

【００１５】さらに、本発明の分散ファイル処理装置において、管理手段は、処理結果保持手段の保持する処理結果に対応し原料ファイル名およびコンテキスト手段における手続き名に基づいて処理結果の有効性を判定する構成を有することを特徴とする。

【００１６】さらに、本発明の分散ファイル処理装置において、処理結果管理手段は、コンテキスト手段による手続き処理の処理実行日時を示す処理日時、コンテキスト手段による手続き処理の実行を要求した要求固有の処理要求識別子の少なくともいずれかに基づいて処理結果の有効性を判定する構成を有することを特徴とする。

【００１７】さらに、本発明の分散ファイル処理装置において、処理結果管理手段は、原料ファイルの更新日時と、コンテキスト手段による手続き処理の処理実行日時を示す処理日時とを比較し、該比較結果に基づいて処理結果の有効性を判定する構成を有することを特徴とする。

【００１８】さらに、本発明の分散ファイル処理装置は、コンテキスト手段が、修飾ファイル名の解釈において、要求処理結果の処理時刻指定子が含まれると判定した場合は、処理結果管理手段は、時刻指定子の示す時間範囲に処理日時が対応するか否かに基づいて、処理日時が対応づけられた処理結果の有効性を判定する構成を有することを特徴とする。

【００１９】さらに、本発明の分散ファイル処理装置において、ネットワークは、複数のコンテキスト手段、該複数のコンテキスト手段に対応して設けられた複数の処理結果保持手段、および該複数の処理結果保持手段に対応して設けられた複数の処理結果管理手段を接続した構成を有し、複数のコンテキスト手段は、それぞれ独立に手続き処理が可能であり、ネットワーク上に出力された処理要求に対応する処理結果の有効性を判定する第１の処理結果管理手段は、該処理要求に指定された原料ファイルおよび手続きに対応する処理結果を保持するネットワーク上の第２の処理結果保持手段を管理する第２の処理結果管理手段に処理結果の有効性についての問い合わせを実行する構成を有することを特徴とする。

【００２０】さらに、本発明の分散ファイル処理装置は、コンテキスト手段における修飾ファイル名の解釈において、処理結果保持手段に保持された処理結果の無効化または破棄いずれかの指示を定めた所定のパラメータが含まれることが解析されたことを条件として、処理結果管理手段は、処理結果保持手段に格納された処理結果の無効化または破棄いずれかの処理を実行する構成を有することを特徴とする。

【００２１】さらに、本発明の分散ファイル処理装置は、処理結果管理手段による処理結果保持手段に格納された処理結果の無効化または破棄いずれかの処理の実行に伴い、コンテキスト手段は、無効化または破棄を行った処理結果に対応する原料ファイルを取得して、新たに処理を実行して処理結果保持手段に保持する構成を有することを特徴とする。

【００２２】さらに、本発明の分散ファイル処理装置において、コンテキスト手段における原料ファイルに対する手続きによる処理は、処理要求タイミングとは独立に非同期的処理として実行するとともに、該手続きによって処理された処理結果には該処理結果に対応する処理結果識別子を生成し、該処理結果識別子を処理結果に対応させて処理結果保持手段に保持する構成としたことを特徴とする。

【００２３】さらに、本発明の分散ファイル処理装置において、コンテキスト手段は、修飾ファイル名の解釈において、処理の要求が非同期的であることを示す指定データが含まれる判定がなされたことを条件として、原料ファイルに対する手続きを処理要求タイミングとは独立に非同期的処理として実行する構成を有することを特徴とする。

【００２４】さらに、本発明の分散ファイル処理装置は、コンテキスト手段が修飾ファイル名の解釈において、処理の要求条件に完全一致しないが、部分一致する処理結果が処理結果保持手段に含まれると判定した場合において、結果出力手段は、処理結果保持手段に含まれる部分一致する処理結果を出力する構成を有することを特徴とする。

【００２５】さらに、本発明の分散ファイル処理方法は、コンピュータをネットワークで接続した分散コンピュータシステムにおける分散ファイル処理方法において、ネットワーク上に接続された装置に保持されたデータファイル原料を識別可能な原料ファイル名を、原料を編集加工する手続きを表す手続き名と、手続きのパラメータと、手続き名を解釈し手続きを動作させるコンピュータおよび動作環境を定めるコンテキスト名とで修飾して構成した修飾ファイル名を解釈して、原料ファイル名と手続き名を取り出し、該取り出した原料ファファイル名に対応する原料ファイルのデータを入力し、取り出した手続き名に対応する手続きを起動して、原料ファイルを該起動した手続きによって処理を行うコンテキスト手段による手続き処理ステップと、手続き処理ステップにおいて起動した手続きによって処理を行った原料ファイルの処理結果を結果保持手段に保持する結果保持ステップと、処理結果保持手段の保持する処理結果の有効性を判定する処理結果有効性判定ステップと、処理結果有効性判定ステップにおいて有効と判定された処理結果であり、処理結果保持手段に保持された処理結果を取り出して出力する結果出力ステップとを有することを特徴とする。

【００２６】さらに、本発明の分散ファイル処理方法において、処理結果有効性判定ステップは、処理結果保持手段の保持する処理結果に対応する原料ファイル名およびコンテキスト手段における手続き名に基づいて処理結果の有効性を判定することを特徴とする。

【００２７】さらに、本発明の分散ファイル処理方法は、コンテキスト手段による手続き処理の処理実行日時、コンテキスト手段による手続き処理の実行を要求した要求固有の処理要求識別子の少なくともいずれか一方に基づいて処理結果の有効性を判定することを特徴とする。

【００２８】さらに、本発明の分散ファイル処理方法において、処理結果有効性判定ステップは、原料ファイルの更新日時と、コンテキスト手段による手続き処理の処理実行日時とを比較し、該比較結果に基づいて処理結果の有効性判定を行うことを特徴とする。

【００２９】さらに、本発明の分散ファイル処理方法において、コンテキスト手段における原料ファイルに対する手続き処理は、処理要求タイミングとは独立に非同期的処理として実行するとともに、該手続きによって処理された処理結果には該処理結果に対応する処理結果識別

子を生成し、該処理結果識別子を処理結果に対応させて処理結果保持手段に保持することを特徴とする。

【００３０】

【発明の実施の形態】本発明の分散ファイル処理システムを構成するネットワーク上の主要な要素について、その概要を図１に示す。図１に示すようにネットワーク上には、ファイルアクセスの実行主体となるクライアント（図１右端）が接続され、また、ファイルの供給主体であり原料ページＡ、Ｂ、Ｃ．．．を保持するサーバ（図１左端）が接続されている。さらに本発明のシステムでは、サーバの保有する原料ページＡ、Ｂ、Ｃ．．．をさまざまな形で加工し、またその加工結果を保持する等、各種の機能を有する処理サーバが存在し、これらが図１ではコンテキスト手段、処理結果保持手段等として示されている。これらの処理サーバの具体的な構成、機能については以下の実施例において詳細に説明する。

【００３１】図１に示す本発明の基本構成における処理の概要について簡単に説明する。まず、ネットワーク上のユーザがWWWクライアントを介してある特定ファイルの加工データを要求する。要求されたファイルの原料ファイルは基本的にWWWサーバの原料ページＡ、Ｂ、Ｃ．．．として保持されているものである。ユーザはこれらの原料ファイルに対して様々な態様で加工した結果データの出力を要求する。例えば原料ページＡに対応するＨＴＭＬ文書の一部を抽出して一つの要約されたＨＴＭＬ文書として出力（メソッドｓｕｍｍａｒｙ）することや、複数のＨＴＭＬ文書を合成して一つのＨＴＭＬ文書として出力（メソッドｍｅｒｇｅ）すること等である。これらの加工処理が図１に示す手続きｆ、ｇ、ｈ．．．の一つの態様である。

【００３２】図１に示す処理結果保持手段が保持するｆ（Ａ）、ｆ（Ｂ）等は原料ページＡ、Ｂに対して、所定の処理を施して得られた結果データである。処理結果保持手段は、WWWクライアントからの要求に応じた処理を行った処理データを保持している場合と保持していない場合がある。また、加工データ、例えばｆ（Ａ）自体は有していても、ユーザのアクセス時には、既にその加工源となった原料ページＡが更新され更新原料ページＡ’として、WWWサーバに保持されている場合もある。このような、様々な場合にWWWクライアントからの要求に柔軟に対応できるシステムを実現するのが本発明のシステムであり、その詳細を以下の実施例において説明する。

【００３３】［実施例１］まず、本発明の分散ファイル処理装置における第１の実施例を説明する。実施例１の分散ファイル処理装置の構成ブロック図を図２に示す。

【００３４】図２に示すように、ネットワークには、原料ファイルをもつWWWサーバ１０００と、コンテキストおよび各種の手続きを実行する処理サーバ１００１，１００２と、ファイルリクエストを出力するWWWブラウザ１００３が接続されている。なお、図２では処理サーバがネットワークに２つ接続されている例を示しているが、これら処理サーバ、WWWサーバ、WWWブラウザの接続数は任意である。

【００３５】WWWサーバ１０００上に原料ファイル１８０１があり、そのＵＲＬは例えば、

ｈｔｔｐ：／／ｈｏｓｔ０／ｐａｇｅ０１．ｈｔｍｌ

であり、これをＵＲＬ１１０１とする。WWWサーバ１０００は、さらに他の原料ページ１８０２、１８０３、１８０４…を有し、これら原料ページ各々に対してＵＲＬ１１０２、ＵＲＬ１１０３、ＵＲＬ１１０４…が対応する。

【００３６】処理サーバ１００１、１００２（ＪＷＳ：Ｊａｖａ　Ｗｅｂ　Ｓｅｒｖｅｒ）上にＪａｖａのクラスであるＥｄｉｔがあり、メソッドとして、Ｅｄｉｔクラスのメソッドであるメソッドｓｕｍｍａｒｙ１２１１、１２２１、ｍｅｒｇｅ１２１２、１２２２他が存在する。

【００３７】この実施例では、図２に示すようにファイルサーバはWWWサーバ１０００で、ファイルはＨＴＭＬフォーマットで記述されたWWWページである。WWWサーバ１０００と、そのクライアントであるWWWブラウザ１００３は、ＨＴＴＰ（ＨｙｐｅｒＴｅｘｔ　Ｔｒａｎｓｆｅｒ　Ｐｒｏｔｏｃｏｌ）でネットワークを介して、WWWページを送受信する。ファイル名はＵＲＬで、例えば、

ｈｔｔｐ：／／ｈｏｓｔ１／ｉｎｄｅｘ．ｈｔｍｌ

などと記される。ＵＲＬのシンタクスは、ＲＦＣ１７３８で規定されている。

【００３８】例えば、処理サーバ１００１のホスト名が、ｈｏｓｔ１であり、ポート番号は８０８０であるとすると、WWWブラウザ１００３は、原料ファイル名ＵＲＬ１１０１を手続き名、パラメータ、コンピュータ名等で修飾した次のような仮想ＵＲＬ１１１１によりリクエストを送出することができる。

ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌ

仮想ＵＲＬについては、以下において詳細に説明する。

【００３９】以下、実施例１について、

「１．仮想ＵＲＬ」

「２．各構成要素の概要」（図２参照）

「３．処理サーバの動作」（図３参照）

「４．コンテキストの動作」（図４参照）

「５．ＵＲＬパーザの動作」（図５参照）

「６．スナップショットマネージャの動作」（図６、図７参照）

「７．レスポンス時の処理」

の順で詳細に説明する。

【００４０】「１．仮想ＵＲＬ」まず、仮想ＵＲＬについて詳細に説明する。原料ファイル名を手続き名、パラメータ、コンピュータ名で修飾したファイル名は、「ＵＲＬ」を手続き名、パラメータ、コンピュータ名で修飾した「仮想ＵＲＬ」として実現する。仮想ＵＲＬの記述を、例をあげて説明する。仮想ＵＲＬは、例えば次のように記述される。

ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌ

【００４１】仮想ＵＲＬの先頭部には、ファイルアクセスのプロトコル、コンテキストの存在する処理サーバのホスト名、ＴＣＰポート番号、コンテキスト起動を指示するパス名が記述され、通常のＵＲＬと同様のシンタクスである。

【００４２】例えば、プロトコルがＨＴＴＰで、ホスト名ｈｏｓｔ１、ポート番号８０８０、パス名がｃｏｎｔｅｘｔ／であるとすれば、仮想ＵＲＬの先頭部は、

ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／

であって、それに続き、コンテキストおよび手続きが指定される。

【００４３】コンテキストはこの実施例ではＪａｖａ言語で記述されたオブジェクトプログラムとして実現され、クラス名で指定する。手続きはＪａｖａで記述されたメソッドで、前記クラス名とメソッド名で指定される。メソッド名は、次の形式で仮想ＵＲＬ中に含める。先頭が文字　？　で始まり、＜ｐｒｏｐｅｒｔｙＮａｍｅ＞＝＜ｐｒｏｐｅｒｔｙＶａｌｕｅ＞の形をなす文字列を文字　＆　で区切って連結する形式であり、ｐｒｏｐｅｒｔｙＮａｍｅは、ｐｒｏｐｅｒｔｙＶａｌｕｅとして記述された値の種類を識別するための名前である。メソッド名は、　ｍｅｔｈｏｄ＝　に続く文字列で指定される。例えば、クラス名Ｅｄｉｔ、メソッド名ｓｕｍｍａｒｙならば、Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙである。

【００４４】パラメータ、原料のＵＲＬも、＜ｐｒｏｐｅｒｔｙＮａｍｅ＞＝＜ｐｒｏｐｅｒｔｙＶａｌｕｅ＞の形式で記述され、　＆　を区切りとしてメソッド名の後ろに追加される。また、この形式ではメソッド名やパラメータや原料のＵＲＬ中の文字　／　，　＆　，　？　，　：　，　＝　はそれぞれ　％２Ｆ　，　％２６，　％３Ｆ　，　％３Ａ　，　％３Ｄ　に変換される。

【００４５】例えば、原料ファイルのＵＲＬが、ｈｔｔｐ：／／ｈｏｓｔ０／ｐａｇｅ０１．ｈｔｍｌで示されるとき、これを仮想ＵＲＬ中に記述すると、＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌに変換される。

【００４６】このように記述される仮想ＵＲＬを解析するＵＲＬパーザ１６１１，１６２１の動作の詳細については、図５を用いて後段で説明する。

【００４７】なお、以下の説明においては、冗長性を回避するため仮想ＵＲＬ等の記述を略記法によって簡略化し記号に置き換えて説明する。仮想ＵＲＬ等と各記号を以下のように対応づける。

【００４８】まず、サーバ名、ポート番号、コンテキスト、クラス名、メソッド名をまとめて、英小文字の記号で表現する。

ｆ：ｈｏｓｔ１上のポート８０８０番のWWWサーバによりアクセスできるコンテキストが起動するクラスＥｄｉｔのメソッド「ｓｕｍｍａｒｙ」を「ｆ」とする。

ｇ：ｈｏｓｔ２上のポート８０８０番のWWWサーバによりアクセスできるコンテキストが起動するクラスＥｄｉｔのメソッド「ｍｅｒｇｅ」を「ｇ」とする。

【００４９】さらに、ＵＲＬ（仮想ＵＲＬも含む）を英大文字の記号で表現する。例えば、

Ａ：「ｈｔｔｐ：／／ｈｏｓｔ０／ｐａｇｅ０１．ｈｔｍｌ」を「Ａ」とする。

Ｂ：「ｈｔｔｐ：／／ｈｏｓｔ０／ｐａｇｅ０２．ｈｔｍｌ」を「Ｂ」とする。

Ｘ：「ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌ」を「Ｘ」とする。

Ｙ：「ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０２．ｈｔｍｌ」を「Ｙ」とする。

Ｚ：「ｈｔｔｐ：／／ｈｏｓｔ２：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｍｅｒｇｅ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ１％３Ａ８０８０％２Ｆｃｏｎｔｅｘｔ％２ＦＥｄｉｔ％３Ｆｍｅｔｈｏｄ％３Ｄｓｕｍｍａｒｙ％２６ｕｒｌ１％３Ｄｈｔｔｐ％２５３Ａ％２５２Ｆ％２５２Ｆｈｏｓｔ０％２５２Ｆｐａｇｅ０１．ｈｔｍｌ＆ｕｒｌ２＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ１％３Ａ８０８０％２Ｆｃｏｎｔｅｘｔ％２ＦＥｄｉｔ％３Ｆｍｅｔｈｏｄ％３Ｄｓｕｍｍａｒｙ％２６ｕｒｌ１％３Ｄｈｔｔｐ％２５３Ａ％２５２Ｆ％２５２Ｆｈｏｓｔ０％２５２Ｆｐａｇｅ０２．ｈｔｍｌ」を「Ｚ」とする。

【００５０】また、手続きに対する原料としてＵＲＬを指示する記述、すなわちＵＲＬが修飾されて仮想ＵＲＬを構成する構造を、関数の表現法と同様に、以下のように表す。例えば、前記Ａ，Ｂ，Ｘ，Ｙ，Ｚ，ｆ，ｇを用いて、Ｘをｆ（Ａ）、Ｙをｆ（Ｂ）、Ｚはｇ（Ｘ，Ｙ）、または、ｇ（ｆ（Ａ），ｆ（Ｂ））と表す。

【００５１】「２．各構成要素の概要」次に本実施例を構成する処理サーバの各要素について、図１と図２を参照しながらその概要を説明する。図１で示すコンテキス

ト手段は、実施例１では次のようなコンテキストオブジェクト（以下、混乱のないかぎりコンテキストと記す）である。図２で示すコンテキスト１４１１，１４２１は、WWWサーバがＨＴＴＰによるファイルの読みだしリクエストとしてＧＥＴや、書込みのリクエストとしてＰＵＴをWWWブラウザ１００３から受けとった時に、WWWサーバから起動される外部プログラムであって、図４で示す動作フローに従って動作する。コンテキストの詳細な動作は図４を用いて後段で詳細に説明する。

【００５２】WWWサーバが外部プログラムを起動する方法は、ＣＧＩ方式やＳｅｒｖｌｅｔ方式（Ｓｅｒｖｌｅｔ方式についてはＪａｖａＷｅｂＳｅｒｖｅｒ ｖｅｒｓｉｏｎ１．１中の文書 Ｔｈｅ Ｊａｖａ Ｓｅｒｖｌｅｔ ＡＰＩ に記されている。）がよく知られている。この実施例では、WWWサーバは、Ｓｕｎ Ｍｉｃｒｏｓｙｓｔｅｍｓ，Ｉｎｃ．のＪａｖａＷｅｂＳｅｒｖｅｒ（以降ＪＷＳとする）とし、コンテキストはＪＷＳが起動する外部プログラムとして、Ｓｅｒｖｌｅｔの形式で実装されている。

【００５３】図１で示す結果保持手段は、実施例１では図２で示すようにＪａｖａのハッシュテーブルとして実現され、処理を行なって生成された処理結果のオブジェクトをスナップショットオブジェクトとして、仮想ＵＲＬをキーとして取り出せる状態で保持する。図２で示す処理サーバ１００１は、スナップシヨット１３１１，１３１２，１３１３，１３１４をハッシュテーブルに保持し、それぞれ仮想ＵＲＬ１１１１、１１１２，１１１３，１１１４をキーとして取り出すことができる。また、処理サーバ１００２は、スナップシヨット１３２１を保持し、仮想ＵＲＬ２１２１をキーとして取り出すことができる。結果出力手段は処理結果のオブジェクトを、ＪＷＳを介してＨＴＴＰプロトコルを用いて出力するプログラムとして実現される。

【００５４】図１で示す結果管理手段は、本実施例では、図２で示すようにスナップショットを保持したハッシュテーブルを管理するＳｅｒｖｌｅｔ形式のプログラムとして実現され、スナップショットマネージャ１５１１，１５２１と呼ばれる。スナップショットマネージャ１５１１，１５２１には、スナップショットの有効性を判定するメソッドが実装されている。有効性を判定するメソッドには、他のＪａｖａプログラムから呼び出され、Ｊａｖａの論理値型の値を返すものと、ＪＷＳがリクエストに応じて起動して、スナップショットが有効かどうかの結果をＨＴＴＰヘッダ中に出力するものがある。スナップショットマネージャの詳細な動作は図６、図７を用いて後段で説明する。

【００５５】「３．処理サーバの動作」次に、処理サーバの動作の詳細について説明する。サーバの動作フローを図３に示す。例として、WWWブラウザ１００３に、ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｍｅｒｇｅ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌ

というＵＲＬをセットして、ファイルを読み込む指示を出力した場合について説明する。

【００５６】WWWブラウザ１００３からの上記のファイル読み込み指示出力に対応して、図３のステップ２０１０においてｈｏｓｔ１というＤＮＳ名を有するホスト名のＪＷＳ、例えば図２で示す処理サーバ１００１は次のようなＨＴＴＰのリクエストを受けとる。

ＧＥＴ／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｍｅｒｇｅ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌＨＴＴＰ／１．０

【００５７】ステップ２０２０においてＪＷＳは、Ｃｏｎｔｅｘｔ．ｃｌａｓｓ（以降、ｃｏｎｔｅｘｔとする）というＪａｖａのサーブレット形式のプログラムが初期化されているかどうか調べる。もし初期化されていなければ、ステップ２０３０においてクラスファイルを読み込んでメモリにロードして初期化し、実行可能な状態とした後、実行を開始する。

【００５８】ステップ２０４０においてＪＷＳはサーブレットｃｏｎｔｅｘｔのメソッドｓｅｒｖｉｃｅを引数のオブジェクトＨｔｔｐＳｅｒｖｌｅｔＲｅｑｕｅｓｔ，ＨｔｔｐＳｅｒｖｌｅｔＲｅｓｐｏｎｓｅを渡して起動する。

【００５９】ステップ２０５０において、コンテキストは、ファイルに対する処理を行い、処理結果をスナップショットオブジェクトとして保持する。引数のオブジェクトＨｔｔｐＳｅｒｖｌｅｔＲｅｓｐｏｎｃｅから、クライアントであるWWWブラウザへの出力ストリームを保持するＳｅｒｖｌｅｔＯｕｔｐｕｔＳｔｒｅａｍを取り出し処理結果を書き込む。ステップ２０５０におけるコンテキストの詳細な動作は図４に示すとおりであり後述する。

【００６０】ステップ２０６０においてＨＴＴＰリクエストのコネクションに対し、ＳｅｒｖｌｅｔＯｕｔｐｕｔＳｔｒｅａｍに書き込まれた内容が出力される。

【００６１】「４．コンテキストの動作」前記図３のステップ２０５０におけるコンテキストの動作を図４に示すコンテキストの動作フローにより詳細に説明する。

【００６２】図４におけるステップ３０１０においてサーブレットｃｏｎｔｅｘｔのメソッドｓｅｒｖｉｃｅが引数のオブジェクトＨｔｔｐＳｅｒｖｌｅｔＲｅｑｕｅｓｔ，ＨｔｔｐＳｅｒｖｌｅｔＲｅｓｐｏｎｓｅを受けとって起動する。

【００６３】ステップ３０２０においてオブジェクトＨｔｔｐＳｅｒｖｌｅｔＲｅｑｕｅｓｔのｇｅｔＲｅｑｕｅｓｔｅｄＵＲＩメソッドにより、仮想ＵＲＬに含まれる文字列、

／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｍｅｒｇ

ｅ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌ
を取り出す。この文字列を先頭部と合成して次の仮想ＵＲＬを再現する。
ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｍｅｒｇｅ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌ

【００６４】ステップ３０３０において仮想ＵＲＬをＵＲＬパーザに渡して解析する。ＵＲＬパーザの動作フローは図５に示す通りであり、この動作については後段で詳細に説明する。ＵＲＬパーザはＶＵＲＬクラスのオブジェクトを生成し、ＵＲＬ中に含まれる、メソッド名、原料のＵＲＬ、パラメータをＶＵＲＬオブジェクトのインスタンス変数にセットして返す。ＶＵＲＬオブジェクトは、仮想ＵＲＬの先頭部、クラス名、メソッド名、パラメータ、原料のＵＲＬ、をインスタンス変数として持つオブジェクトである。

【００６５】ステップ３０４０においてコンテキスト１４１１はスナップショットマネージャ１５１１にＶＵＲＬオブジェクトを渡し、仮想ＵＲＬに対応するスナップショットが有効であるかどうかの答を得る。スナップショットマネージャ１５１１の動作は図６、図７に示すとおりであり、後述する。

【００６６】仮想ＵＲＬに対応する有効なスナップショットが無かったときは、ステップ３０５０において、ＶＵＲＬオブジェクトのインスタンス変数に格納されている原料のＵＲＬに対してＨＴＴＰリクエストを送出し、原料のファイルを取得する。ステップ３０６０においてすべての原料ファイルを取得したかどうかを調べ、すべて取得していなければステップ３０５０に戻る。

【００６７】ステップ３０７０において、ステップ３０５０で取得したファイルを引数として、ＶＵＲＬオブジェクトのインスタンス変数に格納されているクラス名とメソッド名で定まるメソッドを起動しその返り値を処理結果としてオブジェクトを生成する。ステップ３０８０において処理結果のオブジェクトを仮想ＵＲＬをキーにして前記ハッシュテーブルに格納する。ステップ３０９０において処理結果のオブジェクトをＳｅｒｖｌｅｔＯｕｔｐｕｔＳｔｒｅａｍに書き込む。

【００６８】ステップ３０４０において仮想ＵＲＬに対応する有効なスナップショットがあった時は、ステップ３１００においてそのスナップショットオブジェクトを仮想ＵＲＬをキーとしてハッシュテーブルから取り出す。ステップ３１１０においてオブジェクトをＳｅｒｖｌｅｔＯｕｔｐｕｔＳｔｒｅａｍに書き込む。

【００６９】以下に、分散ファイルのプロトコルを用いて前記メソッドが起動され原料ファイルが処理され処理結果が仮想的にファイルとして得られる様子を説明する。前記メソッドはＨＴＭＬ文書の一部を抽出したＨＴＭＬ文書を作成するものとする。ただし、このメソッドはＨＴＭＬ文書を入力し、ＨＴＭＬ文書を出力するプログラムであればどのようなものでも構わない。また、このメソッドの実現の詳細は本発明の動作とは無関係である。

【００７０】図２に示す本実施例に関する全体ブロック図から理解されるように、原料ファイルをもつＷＷＷサーバ１０００と、コンテキストおよび手続きの動作する処理サーバ１００１，１００２と、ＷＷＷブラウザ１００３がネットワークで接続されている。

【００７１】ＷＷＷサーバ１０００上に原料ファイル１８０１があり、そのＵＲＬがｈｔｔｐ：／／ｈｏｓｔ０／ｐａｇｅ０１．ｈｔｍｌであり、これをＵＲＬ１１０１とする。

【００７２】処理サーバ１００１上にＪａｖａのクラスであるＥｄｉｔがあり、前記メソッドはＥｄｉｔクラスのメソッドであるメソッドｓｕｍｍａｒｙ１２１１とする。処理サーバ１００１のホスト名が、ｈｏｓｔ１であり、ポート番号は８０８０である。ＷＷＷブラウザ１００３は、原料ファイル名ＵＲＬ１１０１を修飾した次のような仮想ＵＲＬ１１１１によりリクエストを送出する。

【００７３】ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌ

【００７４】ホスト名がｈｏｓｔ１である処理サーバ１００１が、ＷＷＷブラウザ１００３からの仮想ＵＲＬ１１１１によるＨＴＴＰリクエストを受信すると、コンテキスト１４１１がＪＷＳにより起動される。コンテキスト１４１１は、図４の手順にしたがって動作し、ＵＲＬパーザ１６１１により仮想ＵＲＬ１１１１を解析して仮想ＵＲＬ１１１１を表すＪａｖａオブジェクトＶＵＲＬ１１１１（ｏｂｊ）を生成する。

【００７５】「５．ＵＲＬパーザの動作」この仮想ＵＲＬ１１１１を入力した際のＵＲＬパーザ１６１１の動作を図５に示すフローを用いて説明する。ＵＲＬパーザ１６１１は、図５におけるステップ４０１０において、仮想ＵＲＬ１１１１を受けとって起動する。ステップ４０２０において、仮想ＵＲＬ１１１１を文字　／　を区切り文字として分解し、先頭部ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／と残りの文字列Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌに分ける。

【００７６】ステップ４０３０において、残りの文字列から文字　？　を区切り文字として分解し、クラス名Ｅｄｉｔと残りの文字列ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌに分ける。

【００７７】ステップ４０４０において、残りの文字列から　＆　を区切り文字として＜ｐｒｏｐｅｒｔｙＮａｍｅ＞＝＜ｐｒｏｐｅｒｔｙＶａｌｕｅ＞の形式の文字列を抽出する。メソッド名を含む文字列ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙが取り出される。

【００７８】ステップ４０５０において、取り出された＜ｐｒｏｐｅｒｔｙＮａｍｅ＞＝＜ｐｒｏｐｅｒｔｙＶａｌｕｅ＞の形式の文字列について、ｐｒｏｐｅｒｔｙＮａｍｅが　ｍｅｔｈｏｄ　であるかどうかの判定をおこなう。メソッド名を含む文字列ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙについては、ステップ４０６０にすすむ。

【００７９】ステップ４０６０において、メソッド名を含む文字列ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙから、ｐｒｏｐｅｒｔｙＶａｌｕｅの値であるｓｕｍｍａｒｙを取り出しメソッド名とする。

【００８０】ステップ４０８０において、＜ｐｒｏｐｅｒｔｙＮａｍｅ＞＝＜ｐｒｏｐｅｒｔｙＶａｌｕｅ＞の形式の文字列の処理がすべて終わったかどうか調べ、終っていないので、ステップ４０に戻る。

【００８１】ステップ４０４０において、残りの文字列から　＆　を区切り文字として原料ページ名を含む文字列ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌが取り出される。

【００８２】ステップ４０５０において、ｐｒｏｐｅｒｔｙＮａｍｅが　ｍｅｔｈｏｄ　であるかどうかの判定をおこない、文字列ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌについては、ステップ４０７０にすすむ。

【００８３】ステップ４０７０において文字列ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌから原料ページのＵＲＬであるｈｔｔｐ：／／ｈｏｓｔ０／ｐａｇｅ１．ｈｔｍｌを取り出す。図２におけるＵＲＬ１１０１である。

【００８４】ステップ４０８０において、取り出した＜ｐｒｏｐｅｒｔｙＮａｍｅ＞＝＜ｐｒｏｐｅｒｔｙＶａｌｕｅ＞の形式の文字列の処理がすべて終ったのでステップ４０９０に進む。

【００８５】ステップ４０９０において、仮想ＵＲＬを表すＶＵＲＬクラスのオブジェクトを生成し、先頭部ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／、クラス名Ｅｄｉｔ、メソッド名ｓｕｍｍａｒｙ、原料のＵＲＬであるｈｔｔｐ：／／ｈｏｓｔ０／ｐａｇｅ０１．ｈｔｍｌをインスタンス変数の値としてセットする。以上が仮想ＵＲＬ１１１１を入力した際のＵＲＬパーザ１６１１の動作である。

【００８６】コンテキスト１４１１は、スナップショットマネージャ１５１１にオブジェクトＶＵＲＬ１１１１（ｏｂｊ）を渡し、仮想ＵＲＬ１１１１に対応するスナップショットオブジェクトの有効性を問い合わせる。

【００８７】「６．スナップショットマネージャの動作」スナップショットマネージャ１５１１は、図６、図７に示す動作フローの手順に従って、仮想ＵＲＬ１１１１に対応するスナップショットオブジェクトの有効性をしらべる。

【００８８】スナップショットマネージャ１５１１が、他のスナップショットマネージャからのＨＴＴＰリクエストにより呼ばれるときは、ＪＷＳからＳｅｒｖｌｅｔとして起動され、ステップ５００２から始まるメソッドｓｅｒｖｉｃｅが呼ばれる。このＨＴＴＰリクエストは、有効性を調べるべきスナップショットに対応した仮想ＵＲＬを、ｓｅｒｖｉｃｅメソッドの引数オブジェクトの中に含めるようなＵＲＬでなされる。例えば、サーバｈｏｓｔ２（例えば図２の処理サーバ１００２）のスナップショットマネージャ１５２１に、仮想ＵＲＬ、ｈｔｔｐ：／／ｈｏｓｔ２：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌ
に対応したスナップショットの有効性を問い合わせるリクエストに用いるＵＲＬは、
ｈｔｔｐ：／／ｈｏｓｔ２：８０８０／ｓｅｖｒｌｅｔ／ｓｎａｐｓｈｏｔ？ｕｒｌ＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ２％３Ａ８０８０％２ＦＥｄｉｔ％３Ｆｍｅｔｈｏｄ％３Ｄｓｕｍｍａｒｙ％２６ｕｒｌ１％３Ｄｈｔｔｐ％２５３Ａ％２５２Ｆ％２５２Ｆｈｏｓｔ０％２５２Ｆｐａｇｅ０１．ｈｔｍｌである。
スナップショットマネージャが、ｃｏｎｔｅｘｔサーブレットなどのＪａｖａプログラムから呼ばれるときは、ステップ５０１０から始まるメソッドｃｈｅｃｋＬｏｃａｌが起動される。

【００８９】ステップ５００２においてＨｔｔｐＳｅｒｖｌｅｔＲｅｑｕｅｓｔ，ＨｔｔｐＳｅｒｖｌｅｔＲｅｓｐｏｎｓｅオブジェクトを入力として起動する。ステップ５００４において、ＨｔｔｐＳｅｒｖｌｅｔＲｅｑｕｅｓｔオブジェクトのｇｅｔＱｕｅｒｙＳｔｒｉｎｇ（）メソッドを用いて、有効性を確認すべきスナップショットの仮想ＵＲＬを取り出し、必要なら符号化を解く。ステップ５００６において前記取り出した仮想ＵＲＬをパーズしてＶＵＲＬオブジェクトを生成する。ステップ５００８においてＶＵＲＬオブジェクトを引数としてステップ５０１０から始まるメソッドｃｈｅｃｋＬｏｃａｌを起動する。

【００９０】ステップ５０１０においてＶＵＲＬオブジェクトＶＵＲＬ１１１１（ｏｂｊ）を入力として起動する。ステップ５０２０において、ＶＵＲＬ１１１１（ｏｂｊ）が表す仮想ＵＲＬ１１１１に対応するスナップショットオブジェクトがあるかどうか調べる。これは、ＶＵＲＬ１１１１（ｏｂｊ）のもつインスタンス変数の値から、仮想ＵＲＬ１１１１を構成し、仮想ＵＲＬ１１１１をキーとしてハッシュテーブルに保持されているスナ

ップショットオブジェクトがあるかどうかを調べる。あった時にはステップ５０４０にすすみ、なかった時にはステップ５０３０にすすむ。スナップショットがなかった場合にはステップ５０３０において、スナップショット無効を答えとして、Ｊａｖａの論理値の偽として表現して出力し、呼び出し側に帰る。

【００９１】スナップショットがあった場合にはステップ５０４０において、ＶＵＲＬ１１１１（ｏｂｊ）より原料のＵＲＬであるＵＲＬ１１０１を取り出す。ステップ５０５０において取り出した原料のＵＲＬが、仮想ＵＲＬではあるかどうかを調べ、仮想ＵＲＬならステップ５０６０へ、仮想ＵＲＬで無ければステップ５０７０へ進む。ＵＲＬ１１０１は仮想ＵＲＬではないのでステップ５０７０に進む。

【００９２】ステップ５０６０において、ステップ５０５０において取り出した原料のＵＲＬが仮想ＵＲＬであった場合には、仮想ＵＲＬ中のホスト名で指定されるホストに対して、スナップショットマネージャに有効性を確認させるリクエストをＨＴＴＰで送る。リクエストに用いるＵＲＬは、ＪＷＳに対し、Ｓｅｒｖｌｅｔであるスナップショットマネージャを起動させることを指示するものであり、前記原料のＵＲＬをｓｅｒｖｉｃｅメソッドの引数オブジェクトの中に含めて渡すよう指示する。

【００９３】そのリクエストに対し、レスポンスを返すスナップショットマネージャは、答えを、ＨＴＴＰレスポンスヘッダにヘッダ名　Ｓｎａｐｓｈｏｔ－Ｖａｌｉｄ：に続けて出力し、レスポンスを送出する。レスポンスを受け取るスナップショットマネージャはそのヘッダから文字列　ｔｒｕｅ　または“ｆａｌｓｅ　を抽出して答えとする。

【００９４】ステップ５０７０においてＵＲＬ１１０１の原料ファイルの更新日時を取得する。ステップ５０８０においてＵＲＬ１１０１のファイルの更新日時と仮想ＵＲＬ１１１１に対応するスナップショットオブジェクト１３１１の作成日時とを比較する。ファイルの更新日時が新しい場合は、論理値の答えを真とする。作成日時が新しい場合は答えを偽とする。

【００９５】ステップ５０９０においてＶＵＲＬ１１１１（ｏｂｊ）のインスタンス変数に格納された原料のＵＲＬすべてについてスナップショットが有効かどうかの判定が終了したかどうかを調べ、もし終了していなければステップ５０４０に戻る。ここではＵＲＬ１１０１について結果を得ており、すべて終了したのでステップ５１００に進む。ステップ５１００において、ステップ５０６０または５０８０で得た、原料のＵＲＬについてのスナップショットの有効性の答えの論理積を計算する。ステップ５１３０において、ステップ５１００において計算した論理積の値をスナップショットの有効性の判定結果の答えとして、Ｊａｖａの論理値として表現して呼び出し側に返す。

【００９６】メソッドｓｅｒｖｉｃｅのステップ５１４０において、メソッドｃｈｅｃｋＬｏｃａｌの返り値としてＪａｖａの論理値を受け取る。ステップ５１５０においてＨｔｔｐＳｅｒｖｌｅｔＲｅｓｐｏｎｓｅオブジェクトに対し、ｓｅｔＨｅａｄｅｒメソッドを用い、ＨＴＴＰレスポンスヘッダを表す変数にスナップショットの有効性の答えを格納する。答えは、ヘッダ名を表す文字列　Ｓｎａｐｓｈｏｔ－Ｖａｌｉｄ：　に続き、メソッドｃｈｅｃｋＬｏｃａｌの値が偽なら文字列　ｔｒｕｅ　を、値が偽なら文字列　ｆａｌｓｅ　をヘッダの値として連結する。ステップ５１６０において、メソッドｓｅｒｖｉｃｅを終了し、ＪＷＳが、ＨｔｔｐＳｅｒｖｌｅｔＲｅｓｐｏｎｓｅの値をＨＴＴＰで出力する。

【００９７】以上、図６、図７で示す手順がスナップショットマネージャ１５１１が、仮想ＵＲＬに対応するスナップショットの有効性を答える手順である。

【００９８】「７．レスポンス時の処理」

ａ．スナップショットが有効である場合、

「コンテキストの動作」の欄で説明したように、コンテキスト１４１１は、スナップショットが有効であると判定された場合、仮想ＵＲＬに対応するスナップショットオブジェクト１３１１を取得する（図４、ステップ３１００）。さらに、図３で説明された通りの手順でスナップショットオブジェクト１３１１がＷＷＷブラウザに対するレスポンスとして出力（図３、ステップ２０６０）される。

【００９９】ｂ．スナップショットが無効である場合

コンテキスト１４１１は仮想ＵＲＬ１１１１に対応するスナップショットが無効であると判定された場合、原料ファイル名ＵＲＬ１１０１により、ＷＷＷサーバ１０００に対しファイル取得のリクエストを送出する。このリクエストにより原料ファイル名ＵＲＬ１１０１に対応する原料ページ１８０１を取得（図４、ステップ３０６０）し、ＷＷＷブラウザ１００３からのリクエストＵＲＬ１１１１に指定されたメソッドｓｕｍｍａｒｙ１２１１を起動する。メソッドｓｕｍｍａｒｙ１２１１はＨＴＭＬ文書を入力とし、原料ページの中から特定の部分を抽出してそれを新規に作成したＨＴＭＬページ中に埋め込んだ処理結果のオブジェクトを出力する処理である。コンテキスト１４１１は出力されたオブジェクトをスナップショットオブジェクト１３１１として処理結果オブジェクトとして仮想ＵＲＬ１１１１とペアにしてハッシュテーブルに格納する（図４、ステップ３０８０）。その後、図３で説明された通りの手順でスナップショット１３１１がＷＷＷブラウザ１００３に対するレスポンスとして出力（図３、ステップ２０６０）される。

【０１００】以上の手順で、ＷＷＷブラウザ１００３が送出した仮想ＵＲＬ１１１１リクエスト、

ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ

／Ｅｄｉｔ?ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ&ｕｒｌ１＝ｈｔｔｐ%３Ａ%２Ｆ%２Ｆｈｏｓｔ０%２Ｆｐａｇｅ０１．ｈｔｍｌ
に対する処理が実行される。

【０１０１】［実施例２］上述した実施例１の構成では、図２に示す構成において１つの処理サーバ、例えば処理サーバ１００１、または処理サーバ１００２において手続き処理を実行することが可能な構成について説明した。この第２実施例では、複数の処理サーバが協業して各種処理を実行してWWWブラウザの要求に対応してリスポンスを実行する構成を説明する。

【０１０２】第２の実施例を説明する。第１の実施例で示した図２の構成において、仮想ＵＲＬを原料のＵＲＬとして扱うことにより２つのサーバ（例えば図２の処理サーバ１００１，１００２）における２つの手続きを用いたファイルの処理が行なえる。２つのコンテキストを用いて、ＨＴＭＬ文書からその要約を作成し、それら作成された結果を合成する例をあげて説明する。

【０１０３】以下の説明において、処理サーバ１００１，１００２、コンテキスト１４１１，１４２１、ＵＲＬパーザ１６１１，１６２１、スナップショットマネージャ１５１１，１５２１の動作する手順は実施例１において説明したと同様であるので、それぞれの詳細な記述は省略する。また、ＵＲＬ、仮想ＵＲＬに対する前記の略記法を用いる。

【０１０４】図２に示すように、処理サーバ１００１、１００２にはそれぞれＨＴＭＬ文書中を編集するクラスＥｄｉｔがあり、Ｅｄｉｔクラスには一つのＨＴＭＬ文書の一部を抽出して一つのＨＴＭＬ文書として出力するメソッドｓｕｍｍａｒｙがあり、複数のＨＴＭＬ文書を合成して一つのＨＴＭＬ文書として出力するメソッドｍｅｒｇｅが含まれる。

【０１０５】原料ファイル１８０１，１８０２，１８０３，１８０４がWWWサーバ１０００上にあり、これらのＵＲＬを前記した実施例１中で述べた略記法に従い、「Ａ」，「Ｂ」，「Ｃ」，「Ｄ」とする。サーバ１００１上のコンテキスト１４１１が起動するクラスＥｄｉｔのメソッドｓｕｍｍａｒｙ１２１１による処理を「ｆ」と表す。サーバ１００２上のコンテキスト１４２１が起動するクラスＥｄｉｔのメソッドｍｅｒｇｅ１２２２による処理を「ｇ」と表す。

【０１０６】サーバ１００２が仮想ＵＲＬ：ｇ（ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ））に対するＨＴＴＰリクエストを受信すると、サーバ１００２のコンテキスト１４２１は、サーバ１００２のスナップショットマネージャ１５２１にｇ（ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ））に対応するスナップショットの有効性を問い合わせる。サーバ１００２のスナップショットマネージャ１５２１は、前記図６、図７の手順にしたがって動作し、必要ならばｇ（ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ））から、サーバ１００１への４つの仮想ＵＲＬ：ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ）を取り出して、それらの仮想ＵＲＬについて、それぞれスナップショットマネージャ１５１１に対しスナップショットの有効性を問い合わせるリクエストを送出する。

【０１０７】スナップショットマネージャ１５１１は、それぞれのリクエストを受けとり、リクエストごとに独立に有効性の判定をする。前記図６、図７に示した手順にしたがって判定をおこない、それぞれのリクエストごとにスナップショットの有効性の答えを返す。

【０１０８】サーバ１００２のスナップショットマネージャ１５２１は、図６、図７に示した手順に従い、必要ならば４つの仮想ＵＲＬ：ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ）に対するスナップショットの有効性の答えを受信し、ｇ（ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ））に対応するスナップショットの有効性を計算し、コンテキスト１４２１に対し答えを返す。

【０１０９】もし、スナップショットが無効の場合は、コンテキスト１４２１，１４１１がそれぞれ図４のステップにしたがって動作し、仮想ＵＲＬ：ｇ（ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ））によって起動される処理の結果のオブジェクトを生成し、スナップショットとして格納する。以上のステップにより、WWWブラウザ１００３は、サーバ１０００上の４つの原料ページに対して要約を抽出する処理（ｍｅｒｇｅ）をした結果を、一つのＨＴＭＬ文書中に合成したＨＴＭＬ文書を受信する。

【０１１０】また、処理が完了した後、WWWブラウザ１００３から同じ仮想ＵＲＬによるリクエストが送出された場合に、WWWサーバ１０００上の原料ページ１８０１，１８０２，１８０３，１８０４がいずれも更新されていなければ、すでに作成された処理の結果を出力することができる。また、原料ページのうちの一つ以上が更新されていれば、更新されたページを入力とした処理の結果のスナップショットが無効と判定され、処理が再度起動されてスナップショットが再び生成され、その結果が出力される。

【０１１１】例えば、仮想ＵＲＬ：ｇ（ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ））によるリクエストが送出され、仮想ＵＲＬ：ｇ（ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ））に対応するスナップショット１３２１が生成され、４つの仮想ＵＲＬ：ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ）に対応するスナップショットとして、それぞれスナップショット１３１１，１３１２，１３１３，１３１４が生成されたとする。

【０１１２】その後、ＵＲＬ：Ａで表される原料ファイル１８０１が更新された後に再び仮想ＵＲＬ：ｇ（ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ））によるリクエストが送出された場合、スナップショットマネージャ１５１１は、スナップショット１３１１は無効、スナップ

ショット１３１２，１３１３，１３１４は有効という答えを返す。また、スナップショットマネージャ１５２１はスナップショット１３２１は無効という答えを返す。その結果、コンテキスト１４２１は仮想ＵＲＬ：ｇ（ｆ（Ａ），ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ））によるメソッドｍｅｒｇｅ１２２２の処理をふたたび起動するが、メソッドｍｅｒｇｅ１２２２の入力のうち、３つの仮想ＵＲＬ：ｆ（Ｂ），ｆ（Ｃ），ｆ（Ｄ）による処理は実行されず、それぞれに対応した３つのスナップショット１３１２，１３１３，１３１４が入力される。仮想ＵＲＬ：ｆ（Ａ）による処理は実行され、スナップショット１３１１が更新され、メソッドｍｅｒｇｅ１２２２に入力される。

【０１１３】このように、利用者が原料ファイルの更新を反映した最新の処理結果を得たい場合、最新の処理結果を得るために必要な最小限度の処理を行ない、それ以外は以前の処理の結果のスナップショットを利用するため、リクエストに対する応答時間を短縮することが可能になる。

【０１１４】また、信頼性の低いネットワーク上で複数のサーバ上のコンテキストが起動する処理を組み合わせて一つの処理として利用する際、いずれか一個所以上の通信が失敗すると、最終的な処理結果が得られない。また、そのような場合、全体のすべての処理を再試行すると、やはり一個所以上の通信が失敗する場合には処理結果がえられない。本発明においては、そのような場合に、スナップショットオブジェクトとして一部の処理の結果が保持されるため、再試行により処理結果を得られる確率が増える。

【０１１５】上述した例では、コンテキストが２つの場合を記述したが、３つ以上の場合でも同様の手順にしたがって処理を組み合わせることが可能である。

【０１１６】［実施例３］本発明の第３実施例として、修飾されたＵＲＬの中に含まれた原料ファイル名以外のデータに依存して、スナップショットの有効性を判定する構成例を説明する。システム構成は実施例１と同様、図２に示すものである。

【０１１７】実施例１に述べた構成と基本的には同様のシステム構成を有するが、スナップショットマネージャ１５１１，１５２１がスナップショットの有効性を判定する際、処理サーバ１００１，１００２ごとに特有のルール、あるいは指定された手続きごとに特有のルールに従って特定のオブジェクトを調べ、その結果によりスナップショットの有効性を決定する。

【０１１８】実施例１に述べた文書の合成をする処理を行なう際のメソッドｓｕｍｍａｒｙにおいて、処理結果のＨＴＭＬファイル中に常に処理の行なわれた日付を表示する例を述べる。

【０１１９】ＷＷＷブラウザ１００３を利用する利用者がその時点で得られる最新の情報による処理結果を得たいという要求を持っている場合、処理の行なわれる日時により、日付情報の更新が必要になる。よって日付が変わった場合にはすでに作成され、保持されているスナップショットオブジェクトも更新されるべきであるが、実施例１で説明した手順では、日付情報の参照に関する情報を仮想ＵＲＬ中に記述しない限り、このようなスナップショットオブジェクトが更新されるべきかどうかの判定ができない。

【０１２０】この実施例における構成は図２に示した構成によって実現される。スナップショットマネージャ１５１１，１５２１の動作は図８、図９に示すとおりであり、ステップ５０９２からステップ５１００が実施例１と異なる。

【０１２１】仮想ＵＲＬ１１１１によるリクエストを受けた図２に示すサーバ１００１のスナップショットマネージャ１５１１は、スナップショットの有効性を判定する際に、図８、図９のステップ５０９２において仮想ＵＲＬ１１１１をキーとして、ファイル名のリストをチェックリストの記述されたハッシュテーブルから取得する。ステップ５０９６においてスナップショットマネージャはリスト中のファイルが更新されているかどうか調べる。ステップ５１００において、前記のファイル名を調べた結果および仮想ＵＲＬ１１１１中に記述されたファイルから得た結果をあわせてスナップショットの有効性を判定する。

【０１２２】リスト中の仮想ＵＲＬ１１１１に対応するファイル名としてシステム時刻の日付を記述したファイルを用意することで前記のＷＷＷブラウザ１００３利用者の要求が達成できる。また、この実施例においてチェックリスト中に記述されるファイルは、例えばＣＧＩを用いてプログラムを起動するＷＷＷファイルアクセスのファイル名であってもよい。

【０１２３】本実施例の手順を用いれば、手続き作成者が、チェックリストにファイル名のリストを格納することで、スナップショットを更新するかどうかのルールを定めることが可能となる。手続きにより生成される処理結果が、どの時点で無効になるかを手続き作成者が把握している場合には、この方法を使用することでスナップショットの有効性の判定がより正確になる。

【０１２４】［実施例４］本発明の第４実施例として、処理サーバ１００１，１００２上に保持されているスナップショットの破棄や無効化を指示することができる構成例について説明する。

【０１２５】図２で示す実施例１と同様のシステム構成において、クライアントが送出する仮想ＵＲＬのキュエリ記述部分中にスナップショットの破棄を指示するパラメータを記述する。例えば、図２中に示される仮想ＵＲＬ１１１１は、ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ

０％２Ｆｐａｇｅ０１．ｈｔｍｌであるが、これにスナップショットのコントロールの命令の記述として、ｃｏｎｔｒｏｌｌ＝ｃｌｅａｒというパラメータ記述を追加して、ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｃｏｎｔｒｏｌｌ＝ｃｌｅａｒ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌとする。

【０１２６】実施例１で述べたＵＲＬパーザ１６１１，１６２１にスナップショットの破棄を指示するパラメータを取り出す機能を加えると、ＵＲＬパーザ１６１１，１６２１の動作の流れ図は図１０のようにあらわされる。実施例１のＵＲＬパーザの動作フローとの差異は、ステップ４０５０からステップ４０５２にスナップショットコントロールの命令を取り出す動作が加わった点である。

【０１２７】スナップショットマネージャ１５１１，１５２１の動作フローを図１１、図１２に示す。本実施例において、実施例１のスナップショットマネージャの動作と異なる点は、ステップ５０３２からステップ５０３４のステップである。ステップ５０３２からステップ５０３４において、仮想ＵＲＬから前記スナップショットの破棄を指示するパラメータｃｌｅａｒが取り出された場合、スナップショットマネージャは、指定されたＵＲＬに対応するスナップショットを無効とする動作を実行する。

【０１２８】その後、コンテキストは、実施例１と同様に動作して、原料ファイルの取得と処理の起動を行ない、新たに作成された処理結果のオブジェクトをハッシュテーブルに保持する。または、ＨＴＴＰリクエストヘッダ中に、スナップショットの破棄を指示するパラメータを記述することによっても同様の効果が得られる。コンテキストがＪＷＳから渡されたＨｔｔｐＳｅｒｖｌｅｔＲｅｑｕｅｓｔオブジェクトからＨＴＴＰリクエストのヘッダをとりだし、その中にスナップショットの破棄を指示するパラメータが記述されていた場合は、スナップショットマネージャに対し、指定されたＵＲＬに対応するスナップショットの破棄を指示することによって行う。

【０１２９】実施例１～３においては、スナップショットの無効化は、スナップショットを保持しているサーバのスナップショットマネージャの判定によってなされたが、この実施例においては、他のコンピュータからの指示により処理の結果を無効化することができる。そのため、すでに存在するスナップショットではなく処理を再び行なった結果を得たいという利用者の要求を満たすことができ、新しく生成された処理結果をスナップショットとして保持することができる。

【０１３０】［実施例５］さらに本発明の第５実施例として、処理結果のスナップショットオブジェクトを、それを生成したリクエスト固有の識別子（ＩＤ）と組にして保持することで、特定のリクエストによって生成されたオブジェクトを指定して取得する構成を説明する。本実施例の構成を図１３に示す。図１３において処理サーバ１００１，１００２中のハッシュテーブル中のスナップショット１３１１、１３１２．．．、１３２１．．．には識別子（ＩＤ）が対応して保持されている。

【０１３１】先に説明した実施例１においては、同じ原料に対して同じ手続きを適用するリクエストは同一とみなしていた。すなわち、あるリクエストにより処理を行い、結果をスナップショットオブジェクトとして保持したのち、同一のリクエストを再び受信し処理を再び行った場合は、前記後のリクエストによる処理結果は、前記最初のリクエストにより生成されたスナップショットオブジェクトを置き換えて保持される。

【０１３２】この実施例においては、各リクエストごとに固有の識別子（ＩＤ）が割り当てられ、識別子と対応付けてスナップショットオブジェクトが生成され、保持される。クライアントはリクエストに固有の識別子を指定してそれに対応するスナップショットオブジェクトを取得することができ、また、実施例４で述べた手順を用いれば、指定した識別子に対応するスナップショットオブジェクトを破棄し、新たな処理の結果を同じ識別子に対応付けて保持することができる。

【０１３３】この実施例におけるコンテキストの動作を図１４、図１５に示す。ＨＴＴＰリクエストを受けとったコンテキストは、図１６に従い動作するＵＲＬパーザを用いてＵＲＬを解析してオブジェクトを生成するが、ＵＲＬに識別子（ＩＤ）が含まれていないときはリクエストごとに固有のＩＤを生成して仮想ＵＲＬに付加する。ＩＤは、一例としてリクエストの発生順序の整数値として実現できる。また、別な一例としてシステム時刻の数値として実現できる。仮想ＵＲＬの中に、スナップショットのＩＤを指示するパラメータを付加する。

【０１３４】例えば、実施例１で述べたクラスＥｄｉｔのＨＴＭＬ文書中の一部を抽出するメソッドｓｕｍｍａｒｙによる処理を起動するＵＲＬ１１１１はｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌであるが、これに、ＩＤを付加する記述としてｉｄ＝　の記述に続けて数字を記述する形式を用いてｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｉｄ＝１＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍｌを生成する。

【０１３５】図１４、図１５にコンテキストの動作フローを示す。動作手順について例を挙げて説明する。図１３に示す処理サーバ１００１のコンテキスト１４１１が仮想ＵＲＬ１１１１によるリクエストを受けた場合、ス

テップ３０１０においてサーブレットｃｏｎｔｅｘｔのメソッドｓｅｒｖｉｃｅが引数のオブジェクトＨｔｔｐＳｅｒｖｌｅｔＲｅｑｕｅｓｔ，ＨｔｔｐＳｅｒｖｌｅｔＲｅｓｐｏｎｓｅを受けとって起動する。

【０１３６】ステップ３０２０においてオブジェクトＨｔｔｐＳｅｒｖｌｅｔＲｅｑｕｅｓｔのｇｅｔＲｅｑｕｅｓｔｅｄＵＲＩメソッドにより、仮想ＵＲＬに含まれる文字列　／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｍｅｒｇｅ＆ａｒｇ１＝ｘ＆ａｒｇ２＝ｙ　を取り出す。この文字列を先頭部と合成して仮想ＵＲＬを再構成する.

【０１３７】ステップ３０３０において図１７、図１８に示した手順で動作するＵＲＬパーザ１６１１を用いて仮想ＵＲＬ１１１１を解析してＶＵＲＬ１１１１（ｏｂｊ）を生成する。ステップ３０３２においてＶＵＲＬ１１１１がＩＤの情報を持っているかどうか調べる。仮想ＵＲＬ１１１１にＩＤがついていない場合は、ステップ３０３４に進む。

【０１３８】ステップ３０３４において、ブラウザからのリクエストに対する一意な識別子であるＩＤ１を生成し、ＶＵＲＬ１１１１（ｏｂｊ）に付加する。ステップ３０５０において、仮想ＵＲＬ１１１１より取り出した原料のＵＲＬが仮想ＵＲＬかどうかをしらべる。仮想ＵＲＬであると判定されればステップ３０５２において、もとの仮想ＵＲＬ１１１１と対応づけられたＩＤ１を原料のＵＲＬに付加する。仮想ＵＲＬ１１１１から取り出される原料のＵＲＬは仮想ＵＲＬではないのでステップ３０５４に進む。ステップ３０５４において、ＶＵＲＬ１１１１（ｏｂｊ）のインスタンス変数に格納されている原料のＵＲＬに対してＨＴＴＰリクエストを送出し、原料のファイルを取得する。ステップ３０６０においてすべての原料ファイルを取得したかどうかを調べ、すべて取得していなければステップ３０５０に戻る。

【０１３９】ステップ３０７０において取得したファイルを引数として、ＶＵＲＬオブジェクトのインスタンス変数に格納されているクラス名とメソッド名で定まるメソッドを起動しその返り値を処理結果としてオブジェクトを生成する。ステップ３０８０において処理結果のオブジェクトを仮想ＵＲＬをキーにして前記ハッシュテーブルに格納する。ステップ３０９０において処理結果のオブジェクトをＳｅｒｖｌｅｔＯｕｔｐｕｔＳｔｒｅａｍに書き込む。

【０１４０】クライアントが別のリクエストを送出する際にＩＤを付加した仮想ＵＲＬを用いると、コンテキストは、図１４のステップ３０３２において、ＩＤが付加されているために、３０４０に進む。ステップ３０４０において、コンテキストはスナップショットマネージャにＶＵＲＬオブジェクトを渡し、仮想ＵＲＬ及びＩＤに対応するスナップショットが有効かどうか問い合わせる。

【０１４１】有効なスナップショットがあった場合、ステップ３１００において、そのスナップショットオブジェクトをハッシュテーブルから取り出す。ステップ３１１０においてオブジェクトをＳｅｒｖｌｅｔＯｕｔｐｕｔＳｔｒｅａｍに書き込む。

【０１４２】ＵＲＬパーザは、図１６に示す動作フローに従って動作する。実施例１で説明したＵＲＬパーザの動作フロー（図５参照）との差異は、図１６のステップ４０５０と、ステップ４０５２が加わった点と、ステップ４０９０の出力形態にＩＤが加わった点である。

【０１４３】ステップ４０１０～ステップ４０４０までは実施例１と同様の動作であるので説明を省略する。ステップ４０５０において、取り出された＜ｐｒｏｐｅｒｔｙＮａｍｅ＞＝＜ｐｒｏｐｅｒｔｙＶａｌｕｅ＞の形式の文字列について、ｐｒｏｐｅｒｔｙＮａｍｅが　ｉｄ　であるかどうかの判定をおこなう。判定がＹｅｓであった場合は、ステップ４０５２においてｐｒｏｐｅｒｔｙＶａｌｕｅの値であるＩＤを取り出す。

【０１４４】その後のステップ４０５４～ステップ４０８０は実施例１と同様の動作である。ステップ４０９０では、仮想ＵＲＬを表すＶＵＲＬクラスのオブジェクトを生成し、クラス名Ｅｄｉｔ、メソッド名ｓｕｍｍａｒｙ、ＩＤ、原料のＵＲＬをインスタンス変数の値としてセットする。

【０１４５】スナップショットマネージャは、図１７、図１８にしたがって動作する。実施例１のスナップショットマネージャの動作と異なる点は、ステップ５０２０において、仮想ＵＲＬおよびＩＤが一致するスナップショットオブジェクトが存在するかどうかでスナップショットの有効性を判定する点である。

【０１４６】実施例の細部に関しては、記述した以外の方法を用いても実現可能である。サーバからクライアントにＩＤ情報を伝える方法は、ＨＴＭＬページ中に記述する方法に限らない。ＩＤを付加した仮想ＵＲＬの情報をＨＴＴＰレスポンスのヘッダに含めてクライアントに渡してもよい。

【０１４７】また、ＩＤの生成方法は、時刻とサーバのホスト名によって生成するものであってもよい。クライアントがリクエストを送出する際にＩＤを付加した仮想ＵＲＬを用いることで、特定のリクエストで生成されたオブジェクトを取得することができる。

【０１４８】この実施例で述べた手順により、特定の時点での処理の結果をサーバ上に保持しておき、指定して再度取得することが可能になる。

【０１４９】［実施例６］次に、第６実施例として、サーバ上に保持された複数のスナップショットオブジェクトのうち、生成した時刻の範囲を指定して取得する構成例について説明する。

【０１５０】コンテキストの動作は実施例１と同様の手順であるが、ただし、処理の結果が生成されたときにそ

の時刻を記録し、スナップショットオブジェクトとともに保持する点が異なる。

【０１５１】スナップショットマネージャの動作の流れ図は図１９、図２０に示す。スナップショットマネージャは、図１９、図２０のステップにおいて、仮想ＵＲＬ中にスナップショットオブジェクト生成時刻の範囲が指定されるかどうか調べ、指定されていたらステップ５０３４においてスナップショットオブジェクトの生成時刻を取得し、ステップ５０３６において指定された時刻の範囲にあるかどうかにより判定する。範囲外の場合はステップ５０３８においてスナップショット無効の結果を返す。これらのステップが加わる点が実施例１のスナップショットマネージャの動作と異なる。時刻の範囲が指定されていなければ、実施例１と同様に動作する。

【０１５２】時刻の範囲の指定の例としては、例えばＵＴＣ（Ｃｏｏｒｄｉｎａｔｅｄ　Ｕｎｉｖｅｒｓａｌ　Ｔｉｍｅ）時刻を１９７０年からミリ秒単位で数えた数値で時刻を表現して、例えば８９５８０９１１０００００から８９５８０９１２００００までという時刻の範囲を、仮想ＵＲＬのキュエリ文字列中にｆｔｉｍｅ＝８９５８０９１１０００００＆ｔｔｉｍｅ＝８９５８０９１２００００と記述する。ＵＲＬパーザは前記の動作と同様にしてこの文字列を取り扱い指定時刻を認識、解釈する。

【０１５３】このようにして、生成時刻の正確な指定でなく、ある程度の幅を持った指定をして、スナップショットを利用するか、あるいは処理を起動して新しく結果を生成するかの指定をすることが可能になる。

【０１５４】また、生成時刻に限らず、スナップショットされたオブジェクトの性質を表し、数値表現可能なパラメータをスナップショットオブジェクトと関連付けて保持することにより、同様の手順で、値がある範囲内ならスナップショットを利用する、という条件の指定が行なえる。

【０１５５】［実施例７］次に、第７実施例としてコンテキストによって起動される処理結果が非同期的に生成される構成例を説明する。実施例５と同様の手順で動作するシステムを用いる。ただし、仮想ＵＲＬ中には手続きの呼びだし形式を指定でき、呼び出しが非同期的なものであることを指定できる。指定方法の例としては、ＵＲＬのキュエリ中に、＆ｉｎｖｏｋｅ＝ａｓｙｎｃと指定して、

ｈｔｔｐ：／／ｈｏｓｔ１：８０８０／ｃｏｎｔｅｘｔ／Ｅｄｉｔ？ｍｅｔｈｏｄ＝ｓｕｍｍａｒｙ＆ｉｎｖｏｋｅ＝ａｓｙｎｃ＆ｕｒｌ１＝ｈｔｔｐ％３Ａ％２Ｆ％２Ｆｈｏｓｔ０％２Ｆｐａｇｅ０１．ｈｔｍ

のようなＵＲＬを用いる。

【０１５６】コンテキストは図２１、図２２にしたがって動作する。ステップ３０６０において、仮想ＵＲＬに呼び出しが非同期である指定があるかどうか調べる。非同期の指示があった場合、コンテキストは、ステップ３０６２において原料の取得をおこなって手続きを起動し、別なスレッドで動作させる。次にステップ３０６４においてクライアントに出力する結果オブジェクトに対応するＩＤを含んだ情報を生成する。ＩＤをクライアントに対し出力するには、ＨＴＭＬページ中にハイパーリンクとして記述するなどの方法が利用できる。次にステップ３０６６において生成した情報を出力し、ＨＴＴＰのアクセスが終了してコネクションが切断される。

【０１５７】前記別スレッドの処理が終了後、ステップ３０６８において、結果オブジェクトを、仮想ＵＲＬとＩＤとの組にして保持する。以上のステップ３０６０から３０６８以外は実施例５のコンテキストと同様に動作する。

【０１５８】ＷＷＷクライアント利用者、または他のコンテキストはＩＤを受けとってＨＴＴＰのアクセスを終了し、他の処理をおこなうことができる。その後、実施例５で述べたごとく、仮想ＵＲＬにＩＤを付加してＨＴＴＰリクエストを行なうと、その時点で処理サーバにおけるスナップショット生成処理が終了していれば処理結果を得ることができる。

【０１５９】先に説明した他の実施例においては、呼び出した処理が終了して結果が生成されるまでＷＷＷクライアントとのコネクションが維持され、クライアントはその間、出力データを待機することが要求され、他の作業をできなかったが、この例においては、処理終了までに時間がかかる場合に処理を呼び出した側で並列に別な処理を行うことが可能になる。

【０１６０】処理が実行され生成されたスナップショットはＩＤと対応づけられて処理サーバにおいて保持されており、クライアントはＩＤの付加された新たなＨＴＴＰリクエストによって先に処理を要求したデータをＩＤによる識別により獲得することが可能となる。

【０１６１】［実施例８］次に、本発明の第８実施例として、クライアントがリクエスト時に指定し、仮想ＵＲＬ中に記述された条件を、処理サーバのコンテキストが変更して、クライアントの要求データの代替となりうるスナップショットを探し、そのスナップショットを出力する構成例について説明する。

【０１６２】この実施例構成では、実施例５と同様の手順で動作するシステムを用いる。ただし、指定したＩＤおよび仮想ＵＲＬに対応するスナップショットが存在しないときには、ＩＤが異なるが同一の仮想ＵＲＬに対応するスナップショットを探して有効なスナップショットが存在したときはそれをクライアントに対して出力する構成を持つ。

【０１６３】コンテキストは実施例５と同様に図１４、図１５にしたがって動作する。ステップ３０４０において、スナップショットマネージャに対し仮想ＵＲＬとＩＤに対応するスナップショットの有効性を問い合わせ

る。

【０１６４】スナップショットマネージャの動作は図２３、図２４に示す通りであり、以下に述べる点以外は実施例５のスナップショットマネージャと同じ動作をする。ステップ５０２０において仮想ＵＲＬおよびＩＤに対応するスナップショットオブジェクトがなかったとき、ステップ５０２２に進み、仮想ＵＲＬのみが一致するスナップショットがあるかどうかを調べる。もしなかったときは５０３０にすすみ、スナップショット無効の結果を論理値の偽として返す。スナップショットがあったときは、ステップ５０４０に進み、実施例５と同様にスナップショットの有効性を判定するが、この場合には対象となるスナップショットはステップ５０２２において調べた仮想ＵＲＬのみが一致し、ＩＤについての一致は保証されていないスナップショットである。

【０１６５】図２３、図２４のステップ５０２２におけるスナップショットマネージャの動作は、ここで記述したようなＩＤの一致しないスナップショットを代替として探す動作に限らず、たとえば実施例５において記述した範囲の指定があるような仮想ＵＲＬについて、範囲の指定をなくす、あるいは広げるような動作をさせることも可能である。

【０１６６】前記の実施例５においては、仮想ＵＲＬおよびＩＤが一致しないとスナップショットを利用できなかったが、この実施例の手順によって、利用者の、なるべくＩＤの一致するスナップショットオブジェクトを利用したいが、もし無い場合には代替のオブジェクトを得たいという要求を満たすことが可能になる。

【０１６７】

【発明の効果】上述のように本発明の分散ファイル処理装置および分散ファイル処理方法によれば、原料ファイル名を、原料を編集加工する手続きを表す手続き名と、手続きのパラメータと、手続き名を解釈し手続きを動作させるコンピュータおよび動作環境を定めるコンテキスト名等で修飾して構成した仮想ＵＲＬに基づく処理を行うコンテキスト手段と、コンテキスト手段において処理を行った原料ファイルの処理結果を保持する結果保持手段と、結果保持手段の保持する処理結果の有効性を判定する結果管理手段とを有することにより、結果保持手段の有する処理結果の有効性判定を様々な態様で実行することが可能になり、要求に対して迅速に有効な処理結果を抽出し出力することが可能となる。

【０１６８】さらに、本発明の分散ファイル処理装置および分散ファイル処理方法によれば、原料ファイル名および手続き名等の識別データ、さらに手続き処理の処理実行日時を示す処理日時識別データ、手続き処理の実行を要求した要求固有の処理要求識別データ等に基づいて処理結果の有効性を判定することができ、処理結果に対する信頼度の高い有効性判定が実現される。

【０１６９】さらに、本発明の分散ファイル処理装置および分散ファイル処理方法によれば、複数のコンテキスト手段、結果保持手段、管理手段をネットワークに接続した構成において、処理要求に指定された原料ファイルおよび手続きに対応する処理結果を保持するネットワーク上の他の処理結果保持手段を管理する管理手段に処理結果の有効性についての問い合わせを実行することが可能であり、ネットワーク上の資源の有効な利用が達成される。

【０１７０】さらに、本発明の分散ファイル処理装置および分散ファイル処理方法によれば、コンテキスト手段における原料ファイルに対する手続き処理は、処理要求タイミングとは独立に非同期的処理として実行することが可能であり、処理結果に対応する処理結果識別子を生成して結果保持手段に保持することにより、処理結果の取り出し、出力を正確に実行することが可能となる。

【０１７１】さらに、本発明の分散ファイル処理装置および分散ファイル処理方法によれば、要求された修飾ファイル名の指定と完全一致しないが、部分一致する処理結果が結果保持手段に含まれる場合、部分一致する処理結果を出力する構成としたことにより、クライアント要求により迅速に応答することの可能なシステムが提供される。

【図面の簡単な説明】

【図１】　本発明の分散ファイル処理装置を適用したネットワーク構成を示す図である。

【図２】　本発明の分散ファイル処理装置における実施例１の詳細構成を示す図である。

【図３】　本発明の分散ファイル処理装置、実施例１における処理サーバ全体の動作フローを示す図である。

【図４】　本発明の分散ファイル処理装置、実施例１におけるコンテキストの動作フローを示す図である。

【図５】　本発明の分散ファイル処理装置、実施例１におけるＵＲＬパーザの動作フローを示す図である。

【図６】　本発明の分散ファイル処理装置、実施例１におけるスナップショットマネージャの動作フロー（その１）を示す図である。

【図７】　本発明の分散ファイル処理装置、実施例１におけるスナップショットマネージャの動作フロー（その２）を示す図である。

【図８】　本発明の分散ファイル処理装置、実施例３におけるスナップショットマネージャの動作フロー（その１）を示す図である。

【図９】　本発明の分散ファイル処理装置、実施例３におけるスナップショットマネージャの動作フロー（その２）を示す図である。

【図１０】　本発明の分散ファイル処理装置、実施例４におけるＵＲＬパーザの動作フローを示す図である。

【図１１】　本発明の分散ファイル処理装置、実施例４におけるスナップショットマネージャの動作フロー（その１）を示す図である。

33

【図１２】　本発明の分散ファイル処理装置、実施例４におけるスナップショットマネージャの動作フロー（その２）を示す図である。

【図１３】　本発明の分散ファイル処理装置における実施例５の詳細構成を示す図である。

【図１４】　本発明の分散ファイル処理装置、実施例５におけるコンテキストの動作フロー（その１）を示す図である。

【図１５】　本発明の分散ファイル処理装置、実施例５におけるコンテキストの動作フロー（その２）を示す図である。

【図１６】　本発明の分散ファイル処理装置、実施例５におけるＵＲＬパーザの動作フローを示す図である。

【図１７】　本発明の分散ファイル処理装置、実施例５におけるスナップショットマネージャの動作フロー（その１）を示す図である。

【図１８】　本発明の分散ファイル処理装置、実施例５におけるスナップショットマネージャの動作フロー（その２）を示す図である。

【図１９】　本発明の分散ファイル処理装置、実施例６におけるスナップショットマネージャの動作フロー（その１）を示す図である。

34

【図２０】　本発明の分散ファイル処理装置、実施例６におけるスナップショットマネージャの動作フロー（その２）を示す図である。

【図２１】　本発明の分散ファイル処理装置、実施例７におけるコンテキストの動作フロー（その１）を示す図である。

【図２２】　本発明の分散ファイル処理装置、実施例７におけるコンテキストの動作フロー（その２）を示す図である。

【図２３】　本発明の分散ファイル処理装置、実施例８におけるスナップショットマネージャの動作フロー（その１）を示す図である。

【図２４】　本発明の分散ファイル処理装置、実施例８におけるスナップショットマネージャの動作フロー（その２）を示す図である。

【符号の説明】

１０００　ＷＷＷサーバ

１００１，１００２　処理サーバ

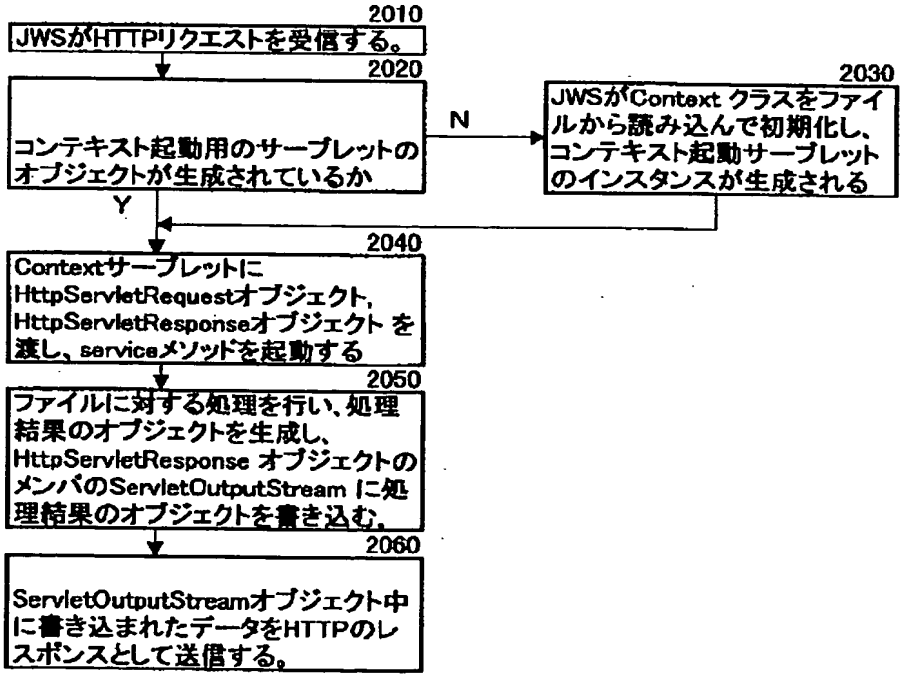
１００３　ＷＷＷブラウザ

１４１１，１４２１　コンテキスト

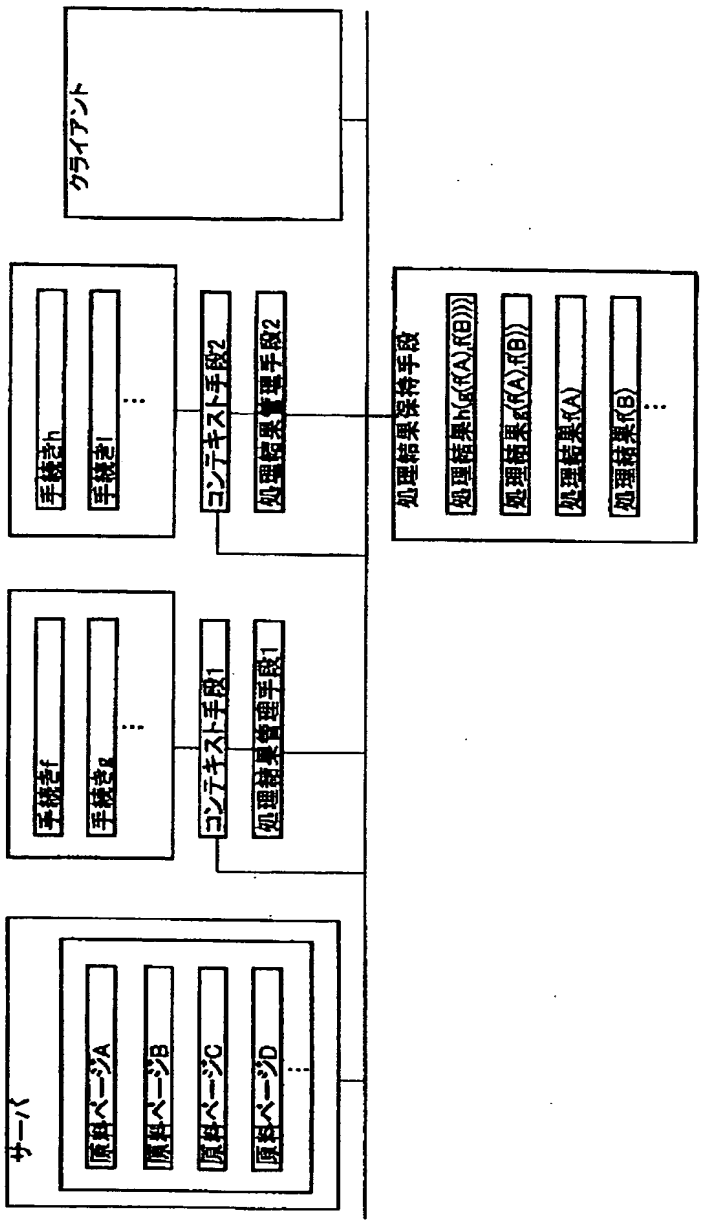
１５１１，１５２１　スナップショットマネージャ

１６１１，１６２１　ＵＲＬパーザ

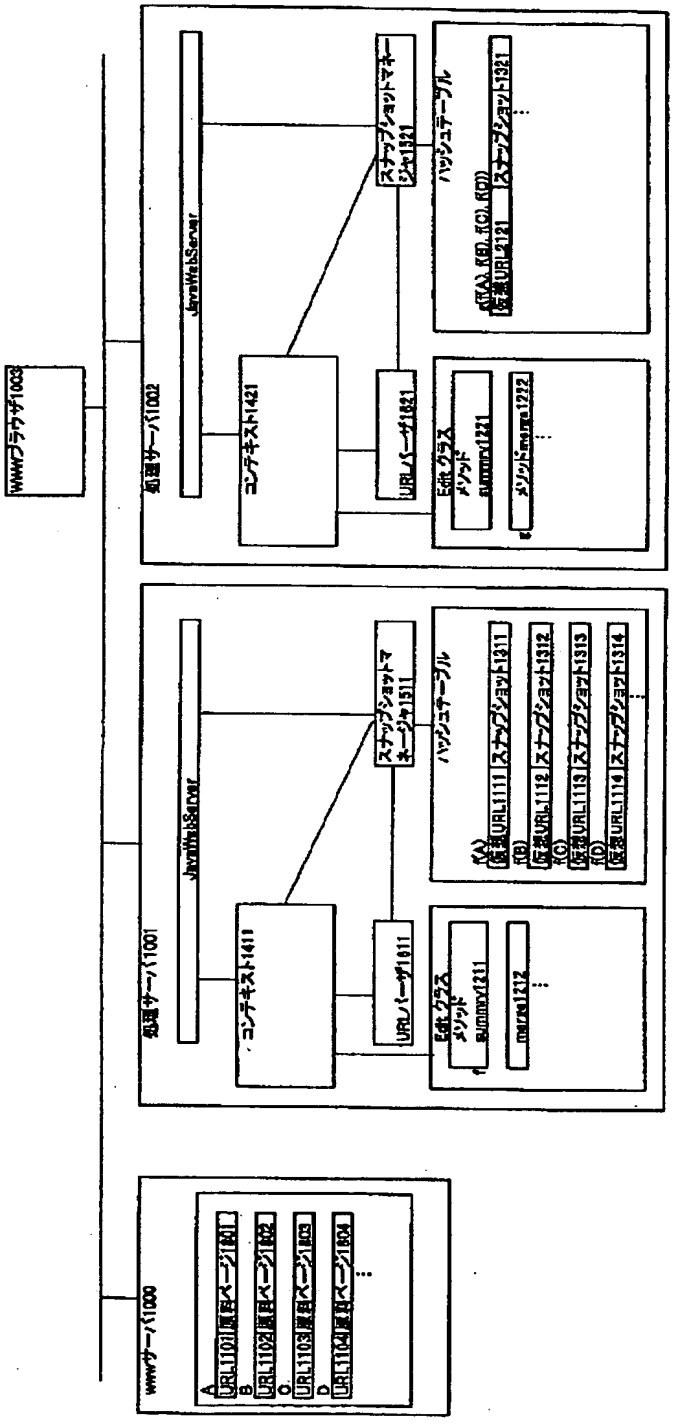
【図３】



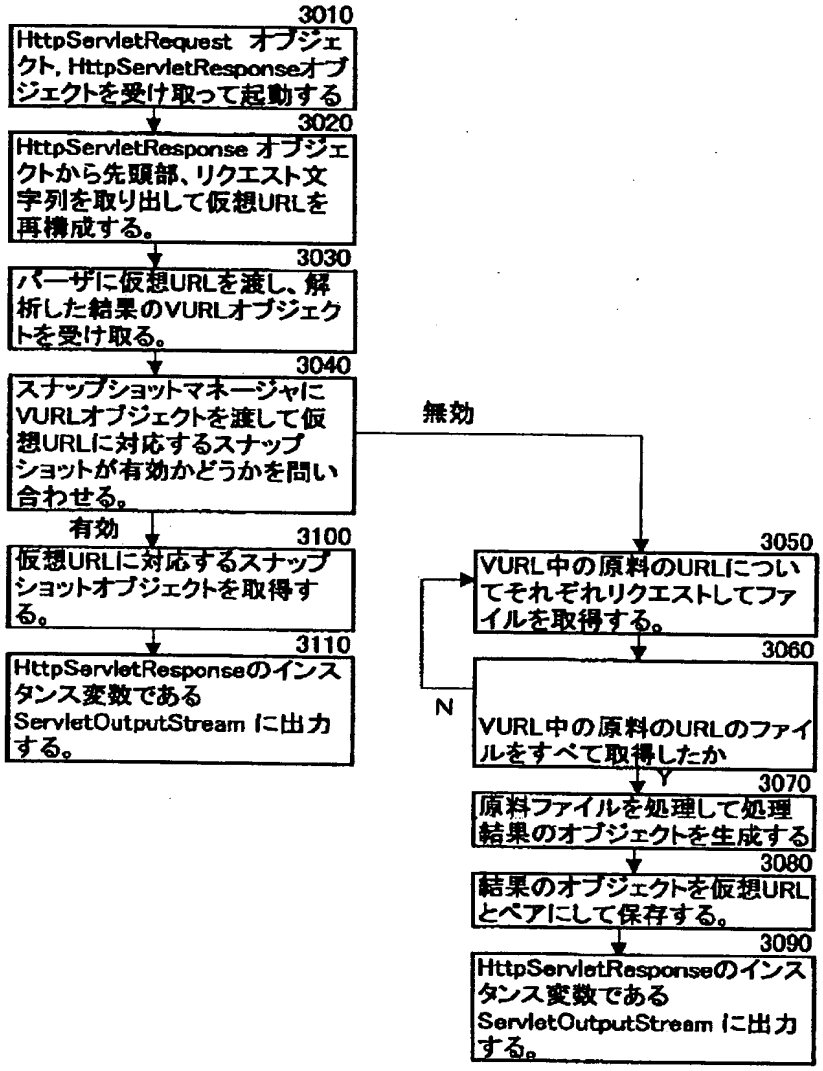
【図１】


クライアント
手続きh
手続きi
コンテキスト手段2
処理結果管理手段2
処理結果保持手段
処理結果h(g(f(A),f(B)))
処理結果g(f(A),f(B))
処理結果f(A)
処理結果f(B)
手続きf
手続きg
コンテキスト手段1
処理結果管理手段1
サーバ
原料ページA
原料ページB
原料ページC
原料ページD

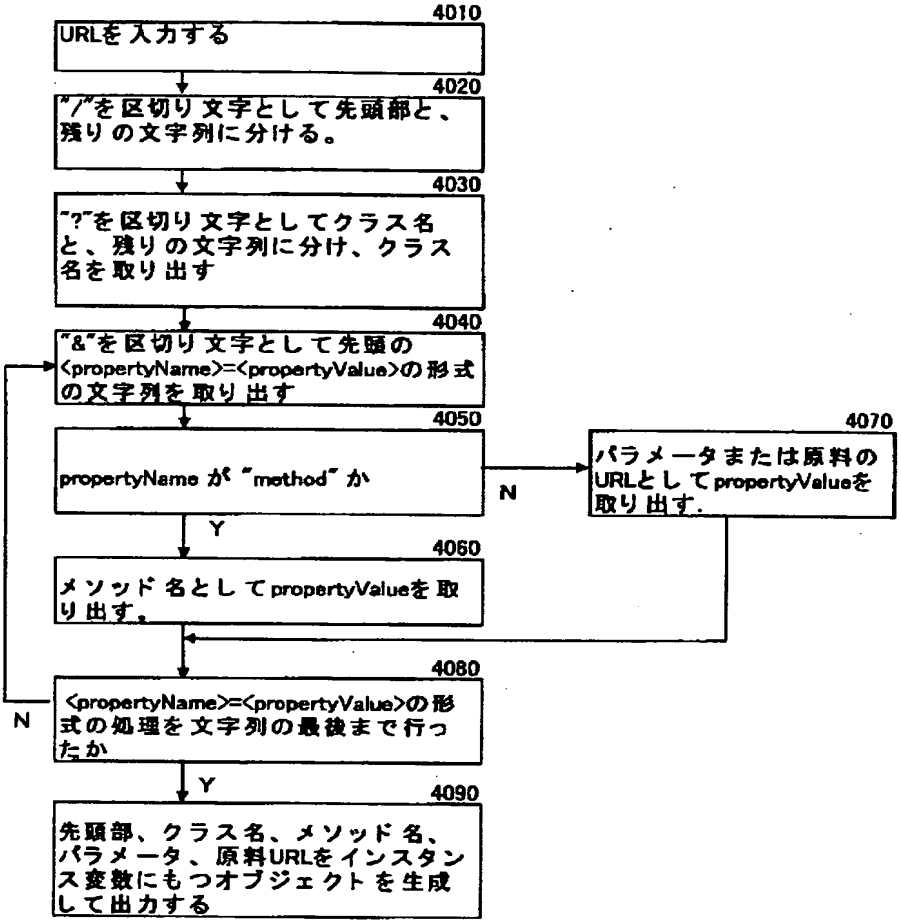
【図２】


wwwプラテナ1003
処理サーバ1002
JavaWebServer
コンテキスト1421
スナップショットマネージャ1321
URLパーザ1321
ハッシュテーブル
スナップショット1321
Edit クラス
メソッド
処理サーバ1001
JavaWebServer
コンテキスト1411
スナップショットマネージャ1311
URLパーザ1311
ハッシュテーブル
スナップショット1311
スナップショット1312
スナップショット1313
スナップショット1314
Edit クラス
メソッド
wwwサーバ1000
URL1101
URL1102
URL1103
URL1104

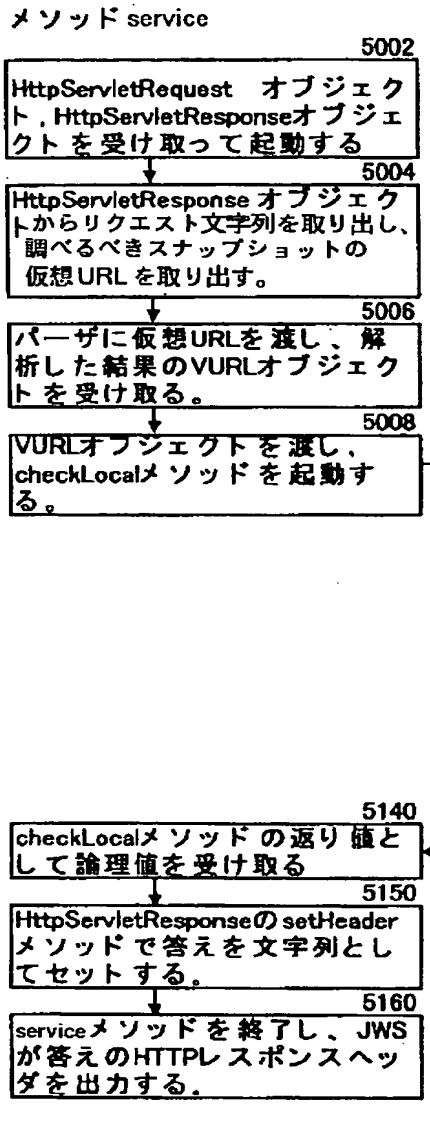
【図４】


3010
HttpServletRequest オブジェクト, HttpServletResponseオブジェクトを受け取って起動する
3020
HttpServletResponse オブジェクトから先頭部、リクエスト文字列を取り出して仮想URLを再構成する。
3030
パーザに仮想URLを渡し、解析した結果のVURLオブジェクトを受け取る。
3040
スナップショットマネージャにVURLオブジェクトを渡して仮想URLに対応するスナップショットが有効かどうかを問い合わせる。
無効
有効
3100
仮想URLに対応するスナップショットオブジェクトを取得する。
3110
HttpServletResponseのインスタンス変数であるServletOutputStream に出力する。
3050
VURL中の原料のURLについてそれぞれリクエストしてファイルを取得する。
3060
VURL中の原料のURLのファイルをすべて取得したか
N
Y
3070
原料ファイルを処理して処理結果のオブジェクトを生成する
3080
結果のオブジェクトを仮想URLとペアにして保存する。
3090
HttpServletResponseのインスタンス変数であるServletOutputStream に出力する。

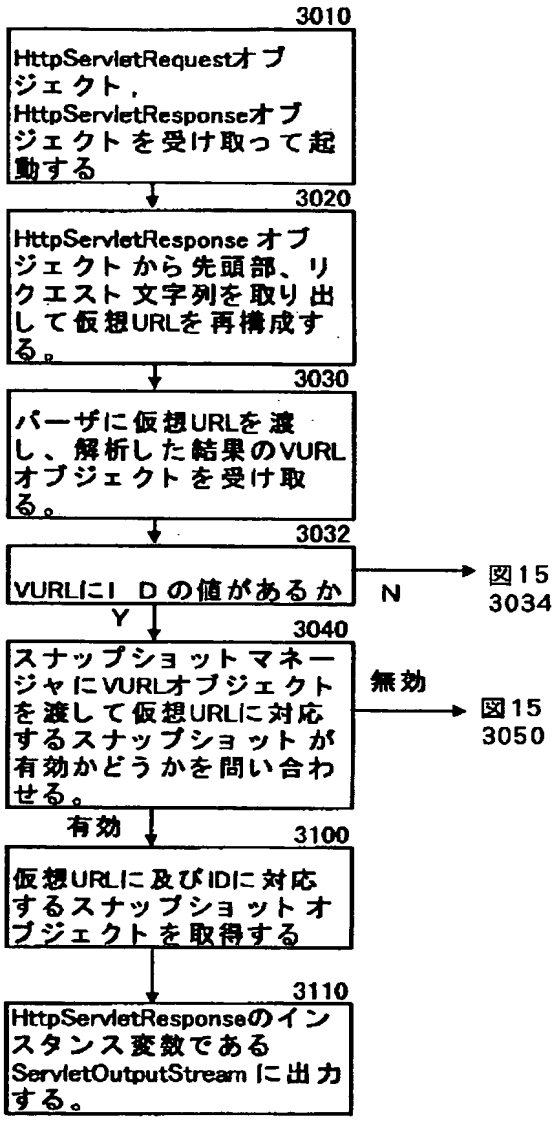
【図５】


4010
URLを入力する
4020
"/"を区切り文字として先頭部と、残りの文字列に分ける。
4030
"?"を区切り文字としてクラス名と、残りの文字列に分け、クラス名を取り出す
4040
"&"を区切り文字として先頭の<propertyName>=<propertyValue>の形式の文字列を取り出す
4050
propertyName が "method" か
N
4070
パラメータまたは原料のURLとしてpropertyValueを取り出す.
Y
4060
メソッド名としてpropertyValueを取り出す。
4080
<propertyName>=<propertyValue>の形式の処理を文字列の最後まで行ったか
N
Y
4090
先頭部、クラス名、メソッド名、パラメータ、原料URLをインスタンス変数にもつオブジェクトを生成して出力する

【図６】


メソッド service
5002
HttpServletRequest オブジェクト，HttpServletResponseオブジェクトを受け取って起動する
5004
HttpServletResponse オブジェクトからリクエスト文字列を取り出し、調べるべきスナップショットの仮想URL を取り出す。
5006
パーザに仮想URLを渡し、解析した結果のVURLオブジェクトを受け取る。
5008
VURLオブジェクトを渡し、checkLocalメソッドを起動する。
図7
5010
5140
checkLocalメソッドの返り値として論理値を受け取る
図7
5150
HttpServletResponseの setHeader メソッドで答えを文字列としてセットする。
5160
serviceメソッドを終了し、JWSが答えのHTTPレスポンスヘッダを出力する。

【図１４】


3010
HttpServletRequestオブジェクト，HttpServletResponseオブジェクトを受け取って起動する
3020
HttpServletResponse オブジェクトから先頭部、リクエスト文字列を取り出して仮想URLを再構成する。
3030
パーザに仮想URLを渡し、解析した結果のVURLオブジェクトを受け取る。
3032
VURLにＩＤの値があるか
N
図15
3034
Y
3040
スナップショットマネージャにVURLオブジェクトを渡して仮想URLに対応するスナップショットが有効かどうかを問い合わせる。
無効
図15
3050
有効
3100
仮想URLに及びIDに対応するスナップショットオブジェクトを取得する
3110
HttpServletResponseのインスタンス変数であるServletOutputStream に出力する。

【図７】

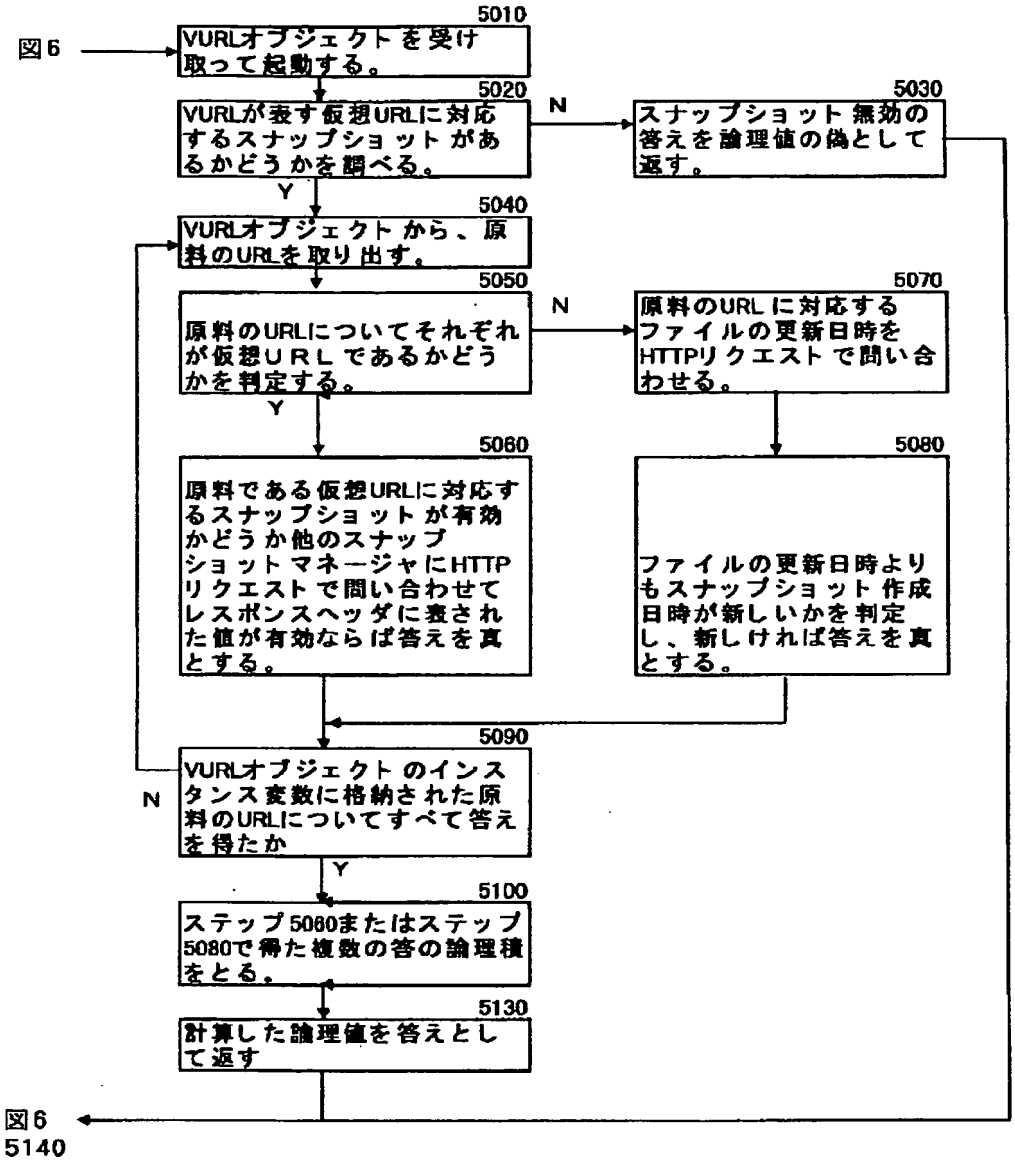
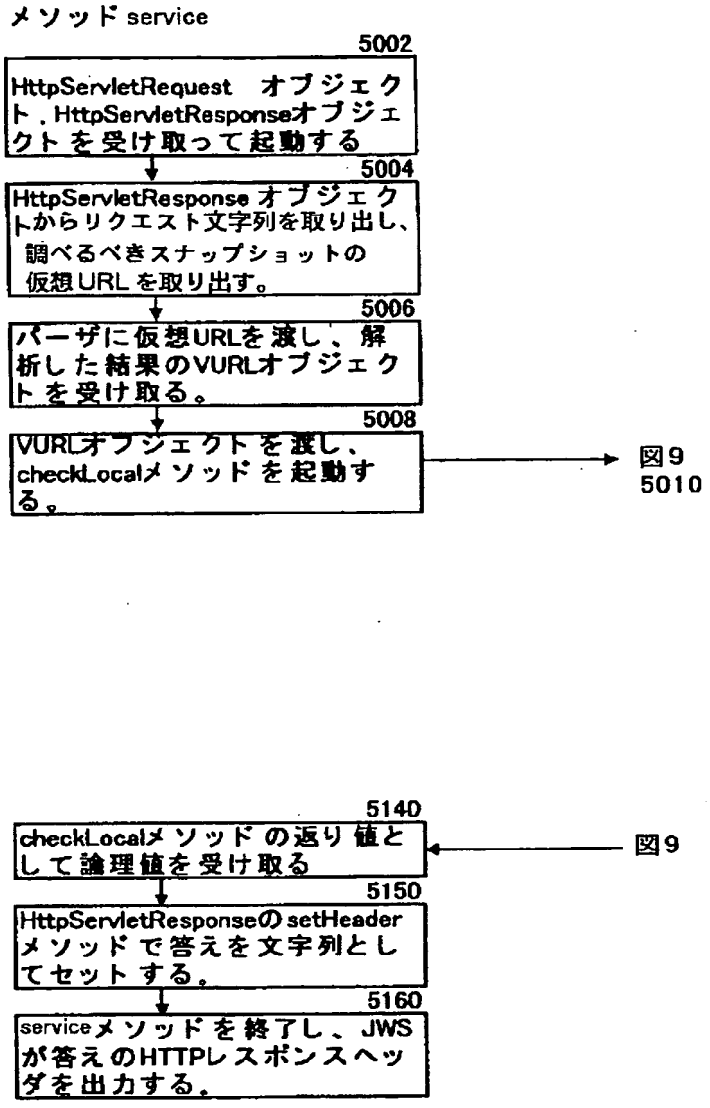


図6
5010
VURLオブジェクトを受け取って起動する。
5020
VURLが表す仮想URLに対応するスナップショットがあるかどうかを調べる。
N
5030
スナップショット無効の答えを論理値の偽として返す。
Y
5040
VURLオブジェクトから、原料のURLを取り出す。
5050
原料のURLについてそれぞれが仮想ＵＲＬであるかどうかを判定する。
N
5070
原料のURLに対応するファイルの更新日時をHTTPリクエストで問い合わせる。
Y
5060
原料である仮想URLに対応するスナップショットが有効かどうか他のスナップショットマネージャにHTTPリクエストで問い合わせてレスポンスヘッダに表された値が有効ならば答えを真とする。
5080
ファイルの更新日時よりもスナップショット作成日時が新しいかを判定し、新しければ答えを真とする。
5090
VURLオブジェクトのインスタンス変数に格納された原料のURLについてすべて答えを得たか
N
Y
5100
ステップ5060またはステップ5080で得た複数の答の論理積をとる。
5130
計算した論理値を答えとして返す
図6
5140

【図８】


メソッド service
5002
HttpServletRequest オブジェクト，HttpServletResponseオブジェクトを受け取って起動する
5004
HttpServletResponse オブジェクトからリクエスト文字列を取り出し、調べるべきスナップショットの仮想URL を取り出す。
5006
パーザに仮想URLを渡し、解析した結果のVURLオブジェクトを受け取る。
5008
VURLオブジェクトを渡し、checkLocalメソッドを起動する。
図9
5010
5140
checkLocalメソッドの返り値として論理値を受け取る
図9
5150
HttpServletResponseのsetHeaderメソッドで答えを文字列としてセットする。
5160
serviceメソッドを終了し、JWSが答えのHTTPレスポンスヘッダを出力する。

【図９】

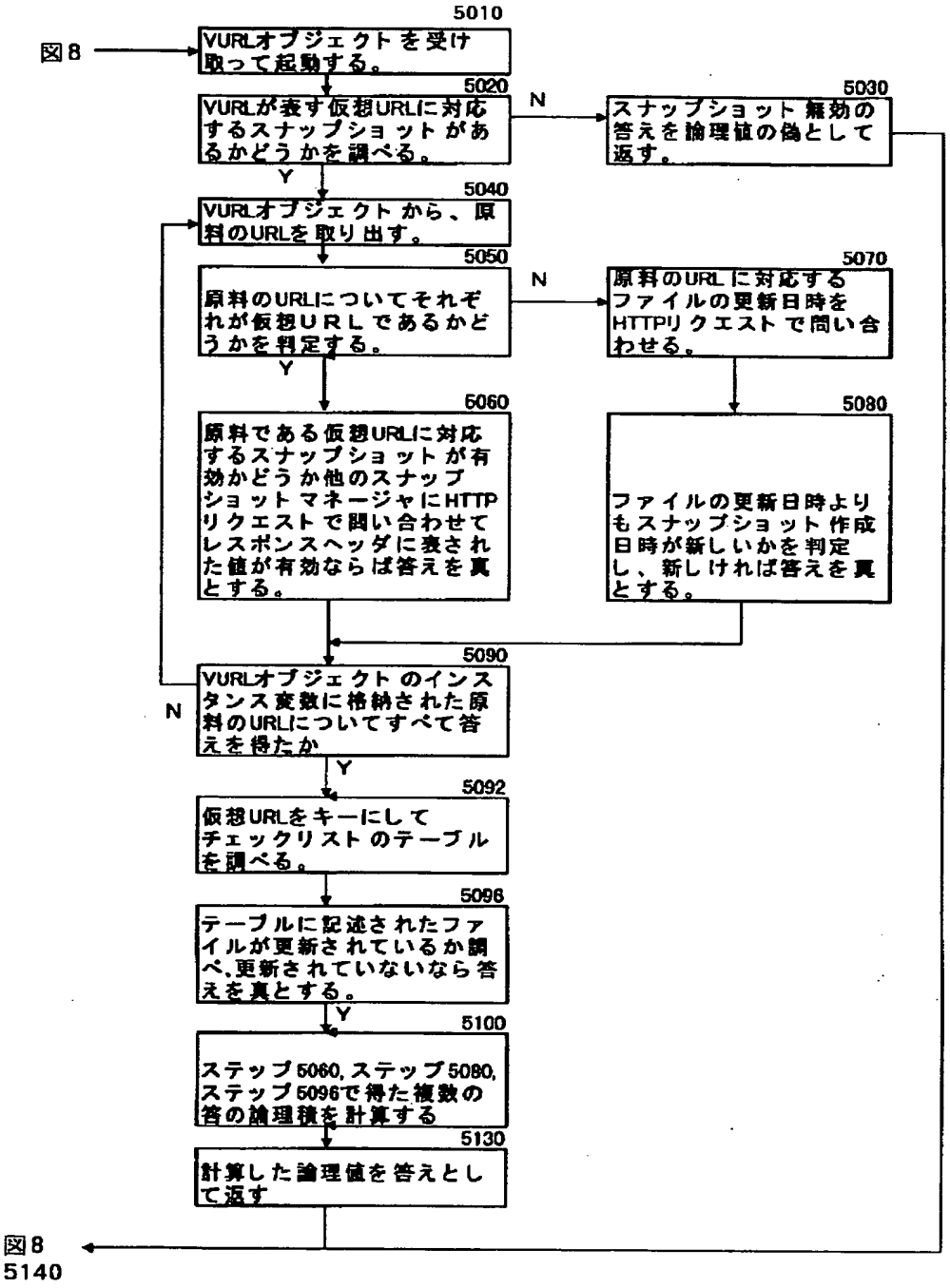
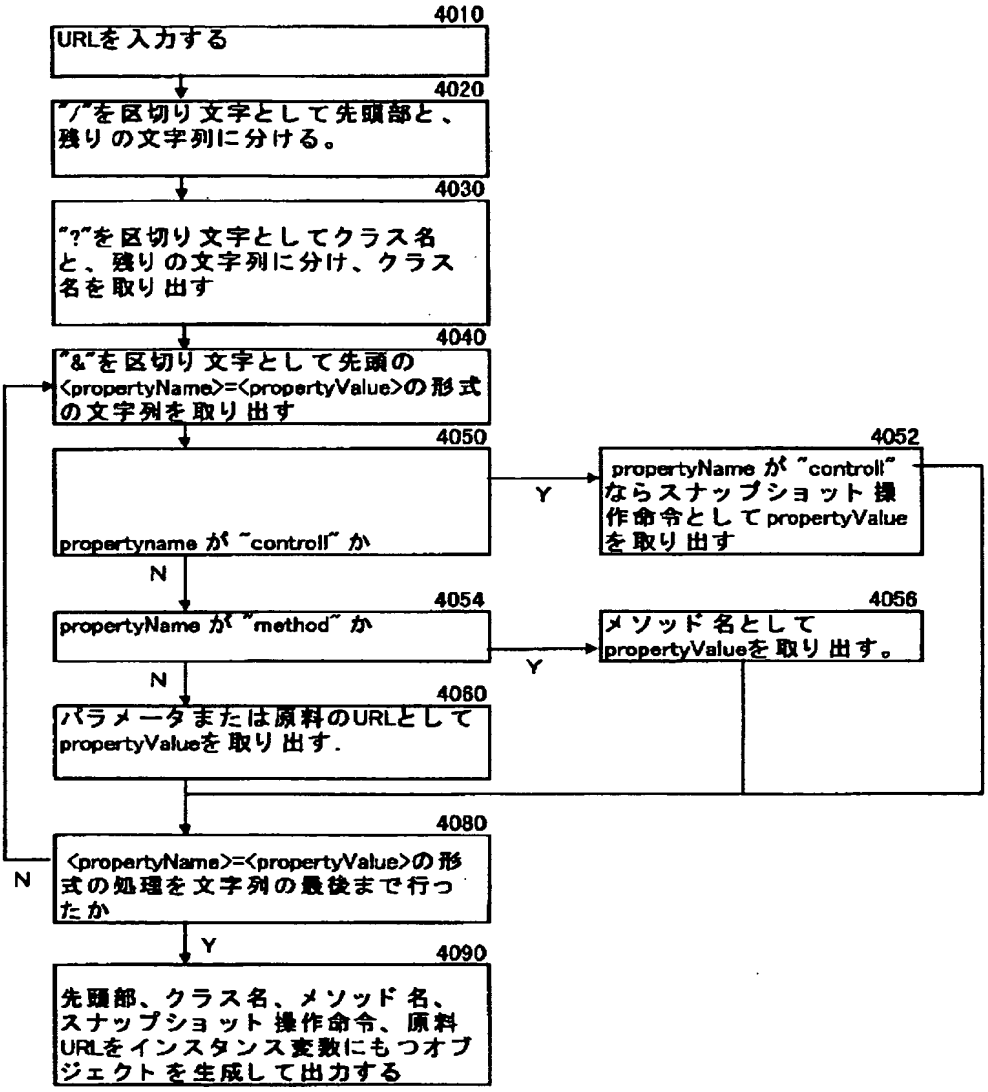
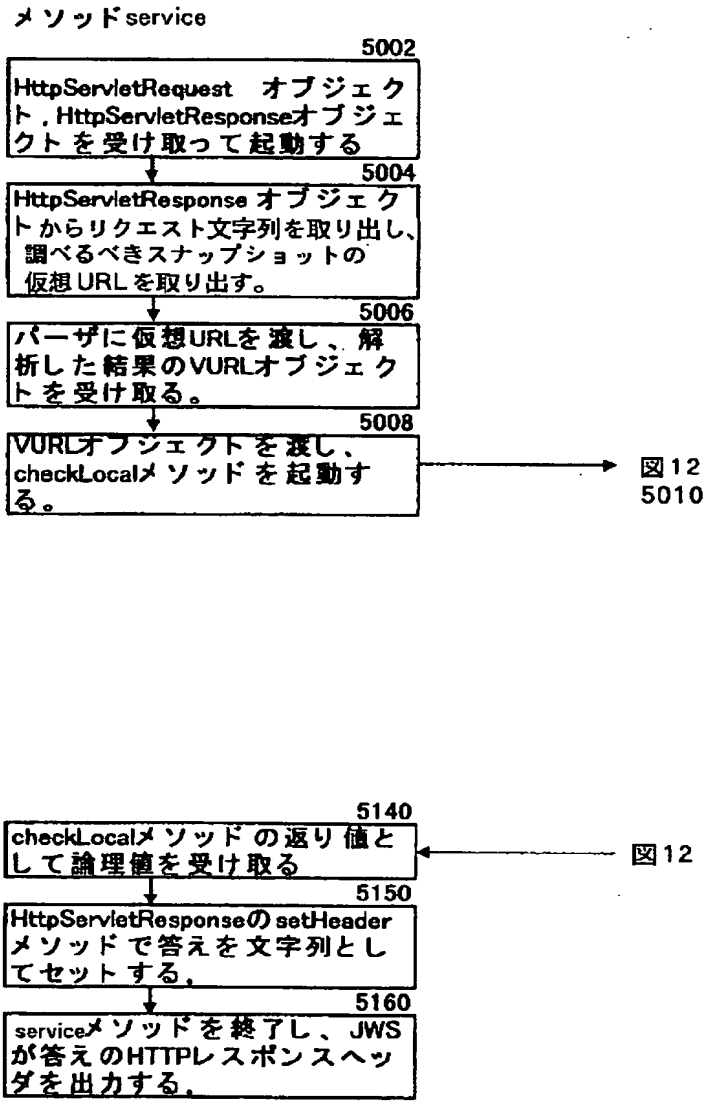


図８
5010
VURLオブジェクトを受け取って起動する。
5020
VURLが表す仮想URLに対応するスナップショットがあるかどうかを調べる。
N
5030
スナップショット無効の答えを論理値の偽として返す。
Y
5040
VURLオブジェクトから、原料のURLを取り出す。
5050
原料のURLについてそれぞれが仮想ＵＲＬであるかどうかを判定する。
N
5070
原料のURLに対応するファイルの更新日時をHTTPリクエストで問い合わせる。
Y
5060
原料である仮想URLに対応するスナップショットが有効かどうか他のスナップショットマネージャにHTTPリクエストで問い合わせてレスポンスヘッダに表された値が有効ならば答えを真とする。
5080
ファイルの更新日時よりもスナップショット作成日時が新しいかを判定し、新しければ答えを真とする。
5090
VURLオブジェクトのインスタンス変数に格納された原料のURLについてすべて答えを得たか
N
Y
5092
仮想URLをキーにしてチェックリストのテーブルを調べる。
5096
テーブルに記述されたファイルが更新されているか調べ、更新されていないなら答えを真とする。
Y
5100
ステップ5060, ステップ5080, ステップ5096で得た複数の答の論理積を計算する
5130
計算した論理値を答えとして返す
図８
5140

【図１０】


4010
URLを入力する
4020
"/"を区切り文字として先頭部と、残りの文字列に分ける。
4030
"?"を区切り文字としてクラス名と、残りの文字列に分け、クラス名を取り出す
4040
"&"を区切り文字として先頭の<propertyName>=<propertyValue>の形式の文字列を取り出す
4050
propertyname が "controll" か
Y
4052
propertyName が "controll" ならスナップショット操作命令としてpropertyValueを取り出す
N
4054
propertyName が "method" か
Y
4056
メソッド名としてpropertyValueを取り出す。
N
4060
パラメータまたは原料のURLとしてpropertyValueを取り出す.
4080
<propertyName>=<propertyValue>の形式の処理を文字列の最後まで行ったか
N
Y
4090
先頭部、クラス名、メソッド名、スナップショット操作命令、原料URLをインスタンス変数にもつオブジェクトを生成して出力する

【図１１】


メソッドservice
5002
HttpServletRequest オブジェクト，HttpServletResponseオブジェクトを受け取って起動する
5004
HttpServletResponse オブジェクトからリクエスト文字列を取り出し、調べるべきスナップショットの仮想URLを取り出す。
5006
パーザに仮想URLを渡し、解析した結果のVURLオブジェクトを受け取る。
5008
VURLオブジェクトを渡し、checkLocalメソッドを起動する。
図12
5010
5140
checkLocalメソッドの返り値として論理値を受け取る
図12
5150
HttpServletResponseのsetHeaderメソッドで答えを文字列としてセットする，
5160
serviceメソッドを終了し、JWSが答えのHTTPレスポンスヘッダを出力する，

【図１２】

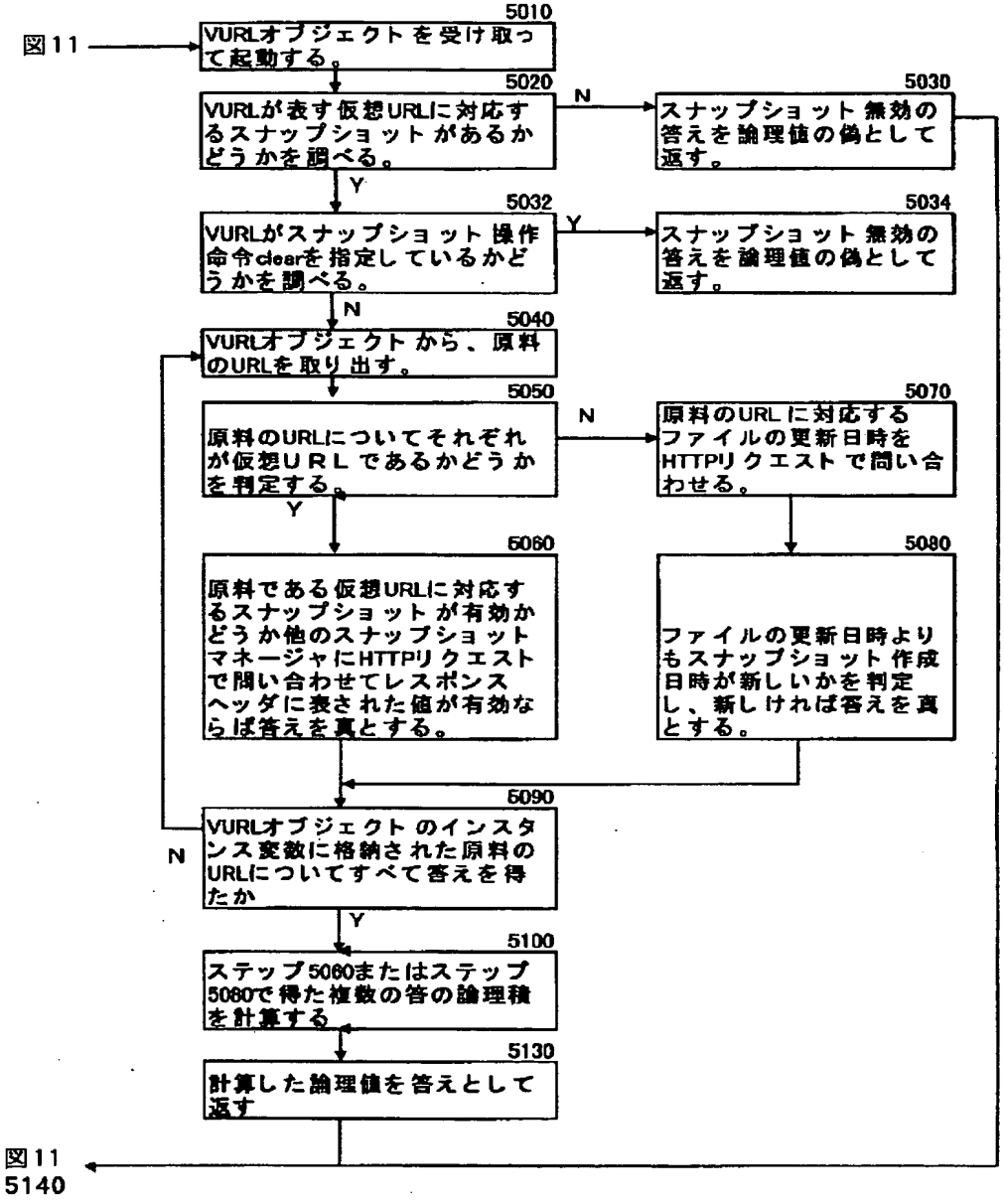
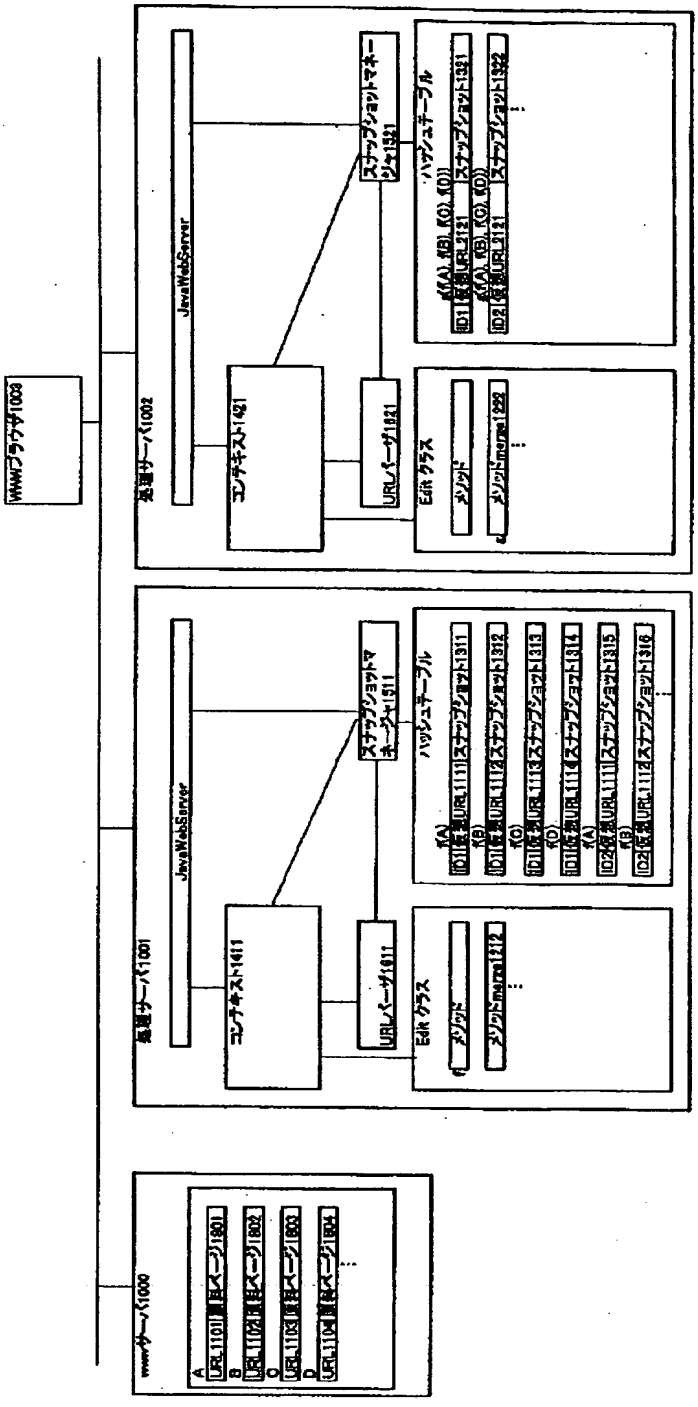


図11
5010
VURLオブジェクトを受け取って起動する。
5020
VURLが表す仮想URLに対応するスナップショットがあるかどうかを調べる。
N
5030
スナップショット無効の答えを論理値の偽として返す。
Y
5032
VURLがスナップショット操作命令clearを指定しているかどうかを調べる。
Y
5034
スナップショット無効の答えを論理値の偽として返す。
N
5040
VURLオブジェクトから、原料のURLを取り出す。
5050
原料のURLについてそれぞれが仮想ＵＲＬであるかどうかを判定する。
N
5070
原料のURLに対応するファイルの更新日時をHTTPリクエストで問い合わせる。
Y
5060
原料である仮想URLに対応するスナップショットが有効かどうか他のスナップショットマネージャにHTTPリクエストで問い合わせてレスポンスヘッダに表された値が有効ならば答えを真とする。
5080
ファイルの更新日時よりもスナップショット作成日時が新しいかを判定し、新しければ答えを真とする。
5090
VURLオブジェクトのインスタンス変数に格納された原料のURLについてすべて答えを得たか
N
Y
5100
ステップ5060またはステップ5080で得た複数の答の論理積を計算する
5130
計算した論理値を答えとして返す
図11
5140

【図１３】


WWWブラウザ1000
高速サーバ1002
JavaWebServer
コンテキスト1421
URLページ1421
スナップショットマネージャ1421
ハッシュテーブル
Edit クラス
メソッド
メソッドname1422
高速サーバ1001
JavaWebServer
コンテキスト1411
URLページ1411
スナップショットマネージャ1411
ハッシュテーブル
Edit クラス
メソッド
メソッドname1412
wwwサーバ1000
A
URL1101 静的ページ1401
B
URL1102 静的ページ1402
C
URL1103 静的ページ1403
D
URL1104 静的ページ1404

【図１５】

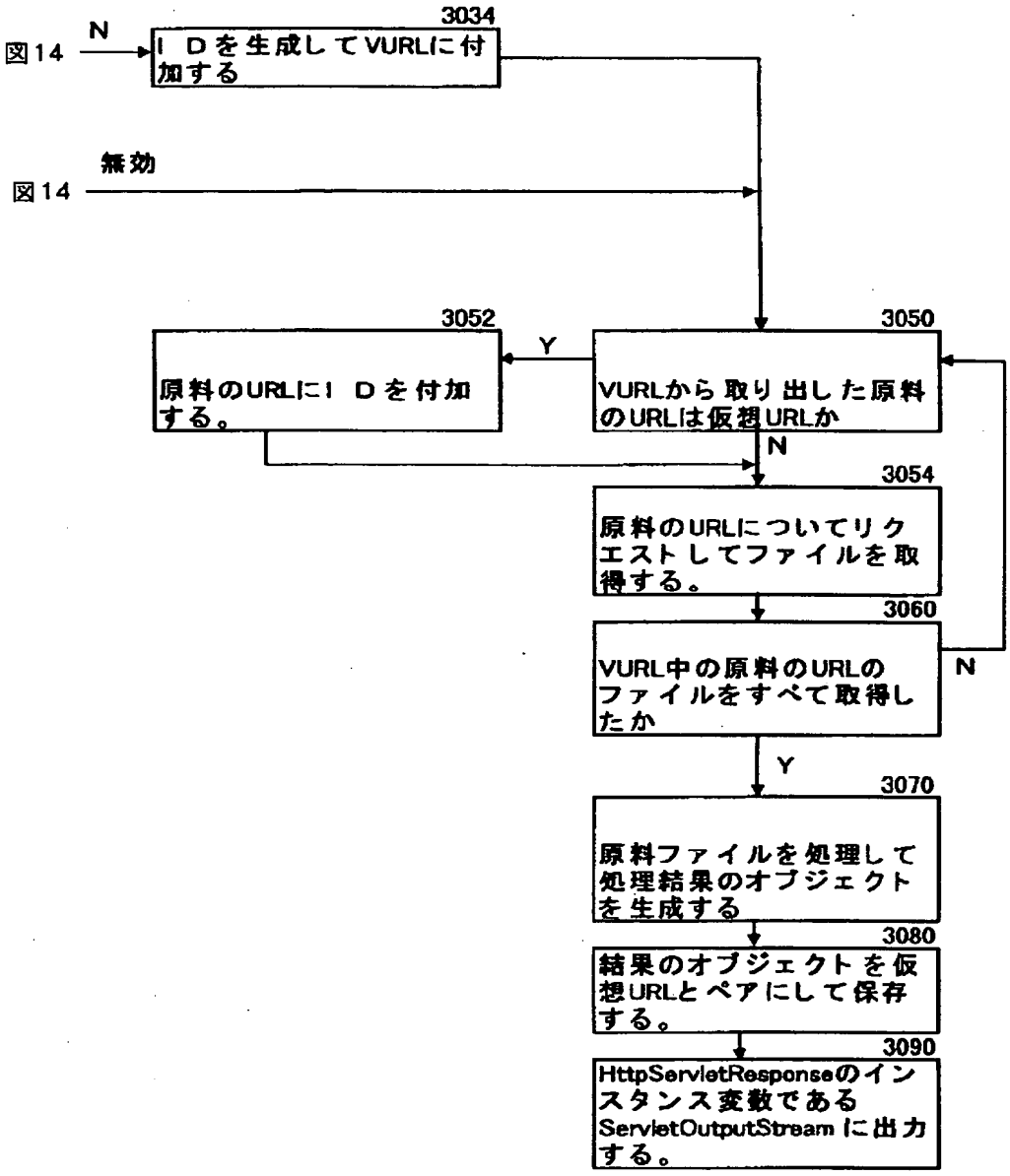
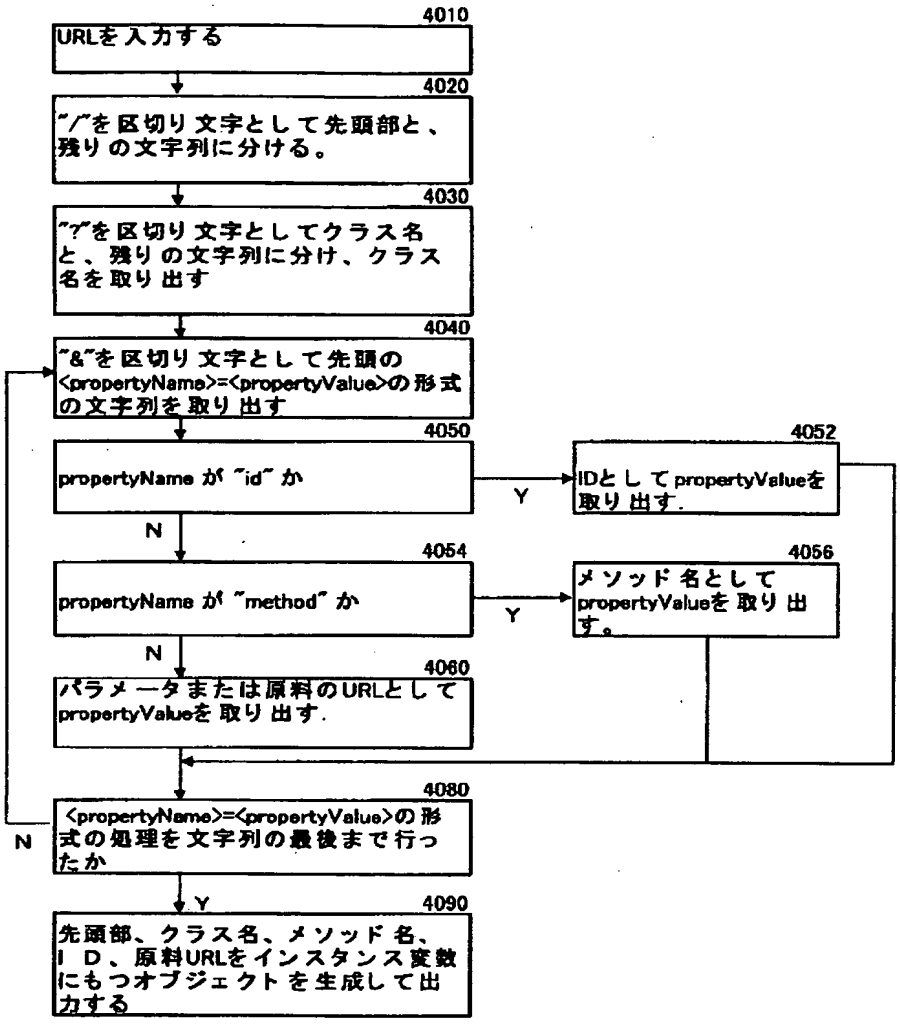
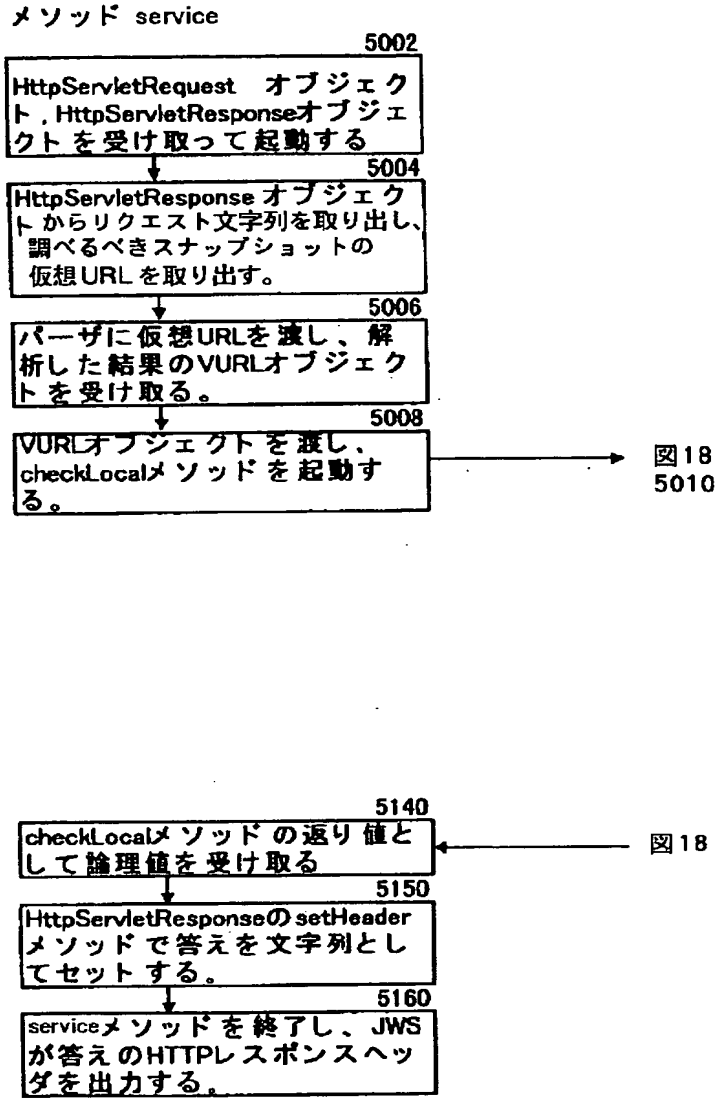


図14
N
3034
ＩＤを生成してVURLに付加する
無効
図14
3052
原料のURLにＩＤを付加する。
Y
3050
VURLから取り出した原料のURLは仮想URLか
N
3054
原料のURLについてリクエストしてファイルを取得する。
3060
VURL中の原料のURLのファイルをすべて取得したか
N
Y
3070
原料ファイルを処理して処理結果のオブジェクトを生成する
3080
結果のオブジェクトを仮想URLとペアにして保存する。
3090
HttpServletResponseのインスタンス変数であるServletOutputStreamに出力する。

【図１６】


4010
URLを入力する
4020
"/"を区切り文字として先頭部と、残りの文字列に分ける。
4030
"?"を区切り文字としてクラス名と、残りの文字列に分け、クラス名を取り出す
4040
"&"を区切り文字として先頭の<propertyName>=<propertyValue>の形式の文字列を取り出す
4050
propertyName が "id" か
Y
4052
IDとしてpropertyValueを取り出す.
N
4054
propertyName が "method" か
Y
4056
メソッド名としてpropertyValueを取り出す。
N
4060
パラメータまたは原料のURLとしてpropertyValueを取り出す.
4080
<propertyName>=<propertyValue>の形式の処理を文字列の最後まで行ったか
N
Y
4090
先頭部、クラス名、メソッド名、ＩＤ、原料URLをインスタンス変数にもつオブジェクトを生成して出力する

【図１７】


メソッド service
5002
HttpServletRequest オブジェクト，HttpServletResponseオブジェクトを受け取って起動する
5004
HttpServletResponseオブジェクトからリクエスト文字列を取り出し、調べるべきスナップショットの仮想URLを取り出す。
5006
パーザに仮想URLを渡し、解析した結果のVURLオブジェクトを受け取る。
5008
VURLオブジェクトを渡し、checkLocalメソッドを起動する。
図18
5010
5140
checkLocalメソッドの返り値として論理値を受け取る
図18
5150
HttpServletResponseのsetHeaderメソッドで答えを文字列としてセットする。
5160
serviceメソッドを終了し、JWSが答えのHTTPレスポンスヘッダを出力する。

【図１８】

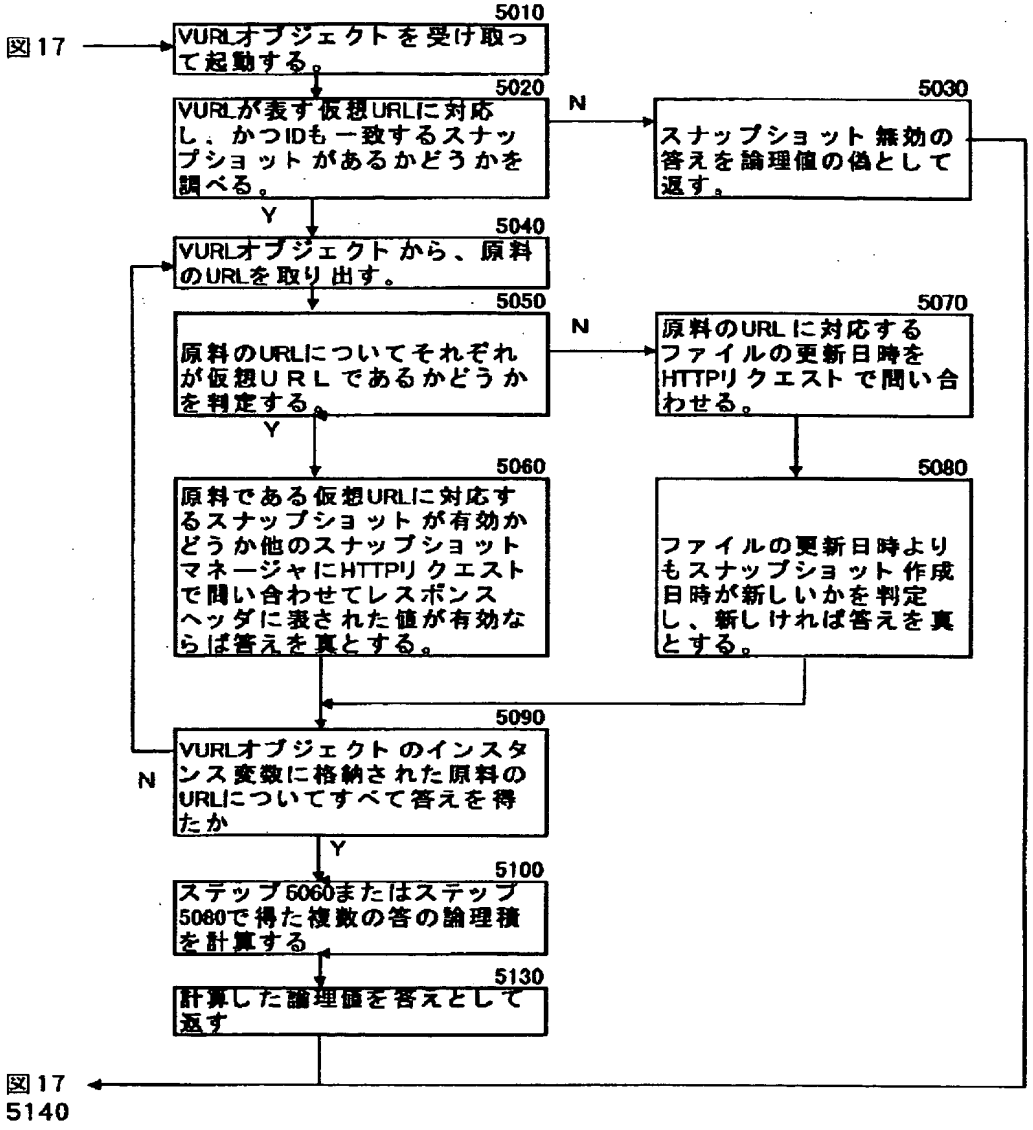
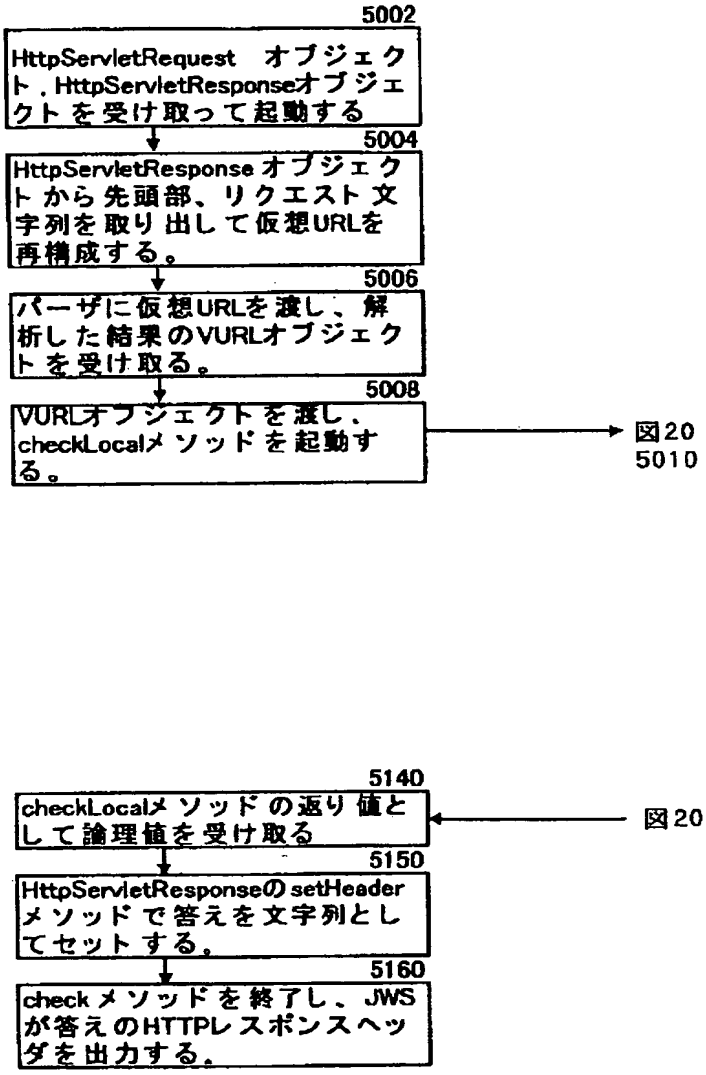


図17
5010
VURLオブジェクトを受け取って起動する。
5020
VURLが表す仮想URLに対応し、かつIDも一致するスナップショットがあるかどうかを調べる。
N
5030
スナップショット無効の答えを論理値の偽として返す。
Y
5040
VURLオブジェクトから、原料のURLを取り出す。
5050
原料のURLについてそれぞれが仮想ＵＲＬであるかどうかを判定する。
N
5070
原料のURLに対応するファイルの更新日時をHTTPリクエストで問い合わせる。
Y
5060
原料である仮想URLに対応するスナップショットが有効かどうか他のスナップショットマネージャにHTTPリクエストで問い合わせてレスポンスヘッダに表された値が有効ならば答えを真とする。
5080
ファイルの更新日時よりもスナップショット作成日時が新しいかを判定し、新しければ答えを真とする。
5090
VURLオブジェクトのインスタンス変数に格納された原料のURLについてすべて答えを得たか
N
Y
5100
ステップ5060またはステップ5080で得た複数の答の論理積を計算する
5130
計算した論理値を答えとして返す
図17
5140

【図１９】


5002
HttpServletRequest オブジェクト，HttpServletResponseオブジェクトを受け取って起動する
5004
HttpServletResponseオブジェクトから先頭部、リクエスト文字列を取り出して仮想URLを再構成する。
5006
パーザに仮想URLを渡し、解析した結果のVURLオブジェクトを受け取る。
5008
VURLオブジェクトを渡し、checkLocalメソッドを起動する。
図20
5010
5140
checkLocalメソッドの返り値として論理値を受け取る
図20
5150
HttpServletResponseのsetHeaderメソッドで答えを文字列としてセットする。
5160
checkメソッドを終了し、JWSが答えのHTTPレスポンスヘッダを出力する。

【図２０】

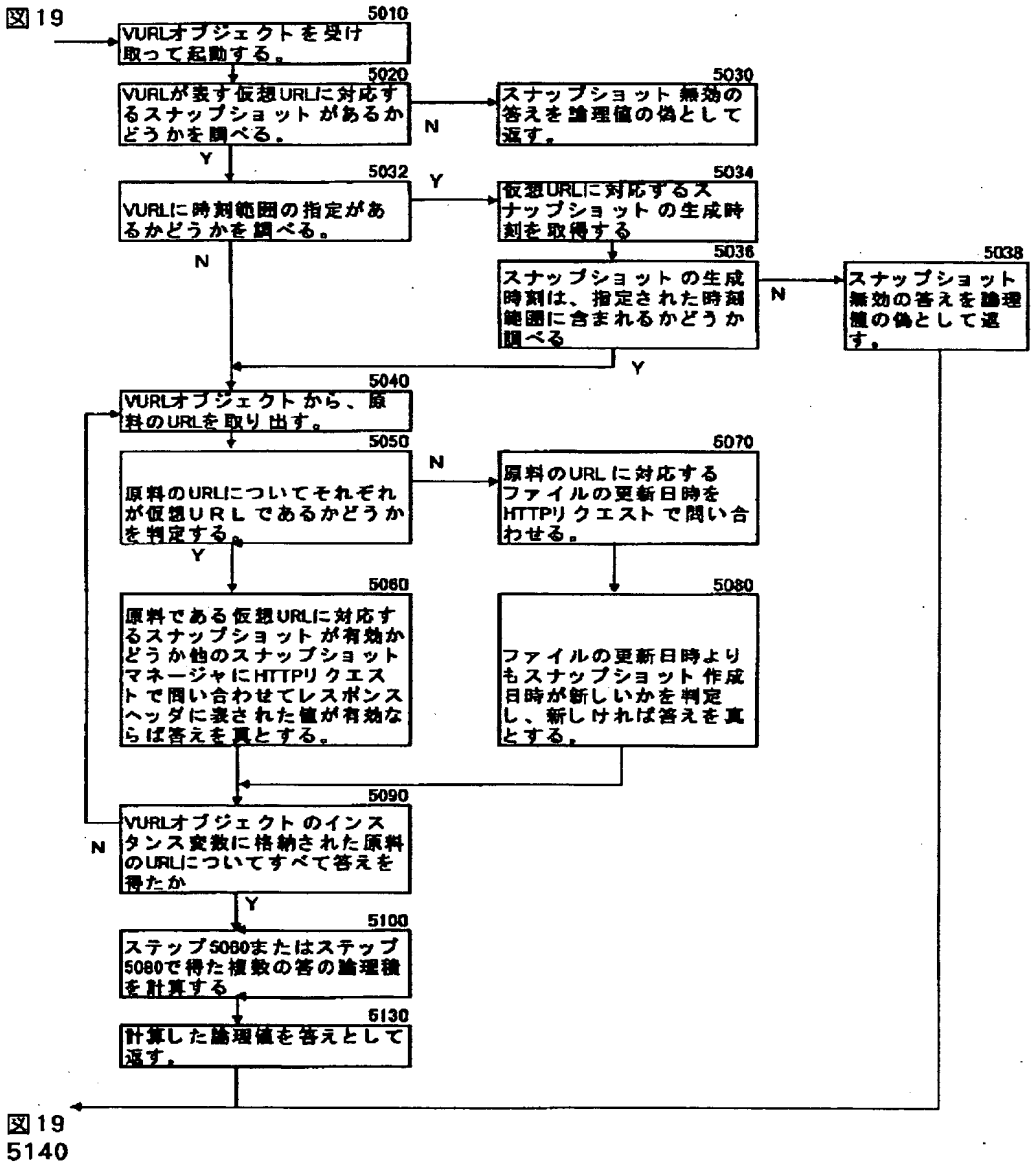
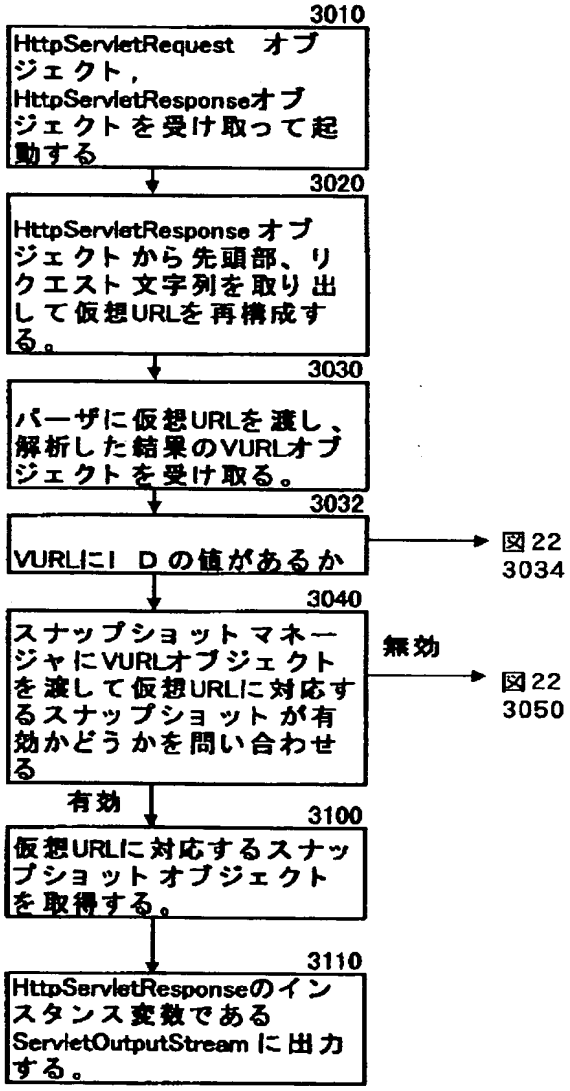


図19
5010
VURLオブジェクトを受け取って起動する。
5020
VURLが表す仮想URLに対応するスナップショットがあるかどうかを調べる。
N
5030
スナップショット無効の答えを論理値の偽として返す。
Y
5032
VURLに時刻範囲の指定があるかどうかを調べる。
Y
5034
仮想URLに対応するスナップショットの生成時刻を取得する
N
5036
スナップショットの生成時刻は、指定された時刻範囲に含まれるかどうか調べる
N
5038
スナップショット無効の答えを論理値の偽として返す。
Y
5040
VURLオブジェクトから、原料のURLを取り出す。
5050
原料のURLについてそれぞれが仮想ＵＲＬであるかどうかを判定する。
N
5070
原料のURLに対応するファイルの更新日時をHTTPリクエストで問い合わせる。
Y
5060
原料である仮想URLに対応するスナップショットが有効かどうか他のスナップショットマネージャにHTTPリクエストで問い合わせてレスポンスヘッダに表された値が有効ならば答えを真とする。
5080
ファイルの更新日時よりもスナップショット作成日時が新しいかを判定し、新しければ答えを真とする。
5090
VURLオブジェクトのインスタンス変数に格納された原料のURLについてすべて答えを得たか
N
Y
5100
ステップ5060またはステップ5080で得た複数の答の論理積を計算する
5130
計算した論理値を答えとして返す。
図19
5140

【図２１】


3010
HttpServletRequest オブジェクト，HttpServletResponseオブジェクトを受け取って起動する
3020
HttpServletResponse オブジェクトから先頭部、リクエスト文字列を取り出して仮想URLを再構成する。
3030
パーザに仮想URLを渡し、解析した結果のVURLオブジェクトを受け取る。
3032
VURLにＩＤの値があるか
図22 3034
無効
3040
スナップショットマネージャにVURLオブジェクトを渡して仮想URLに対応するスナップショットが有効かどうかを問い合わせる
図22 3050
有効
3100
仮想URLに対応するスナップショットオブジェクトを取得する。
3110
HttpServletResponseのインスタンス変数であるServletOutputStreamに出力する。

【図２２】

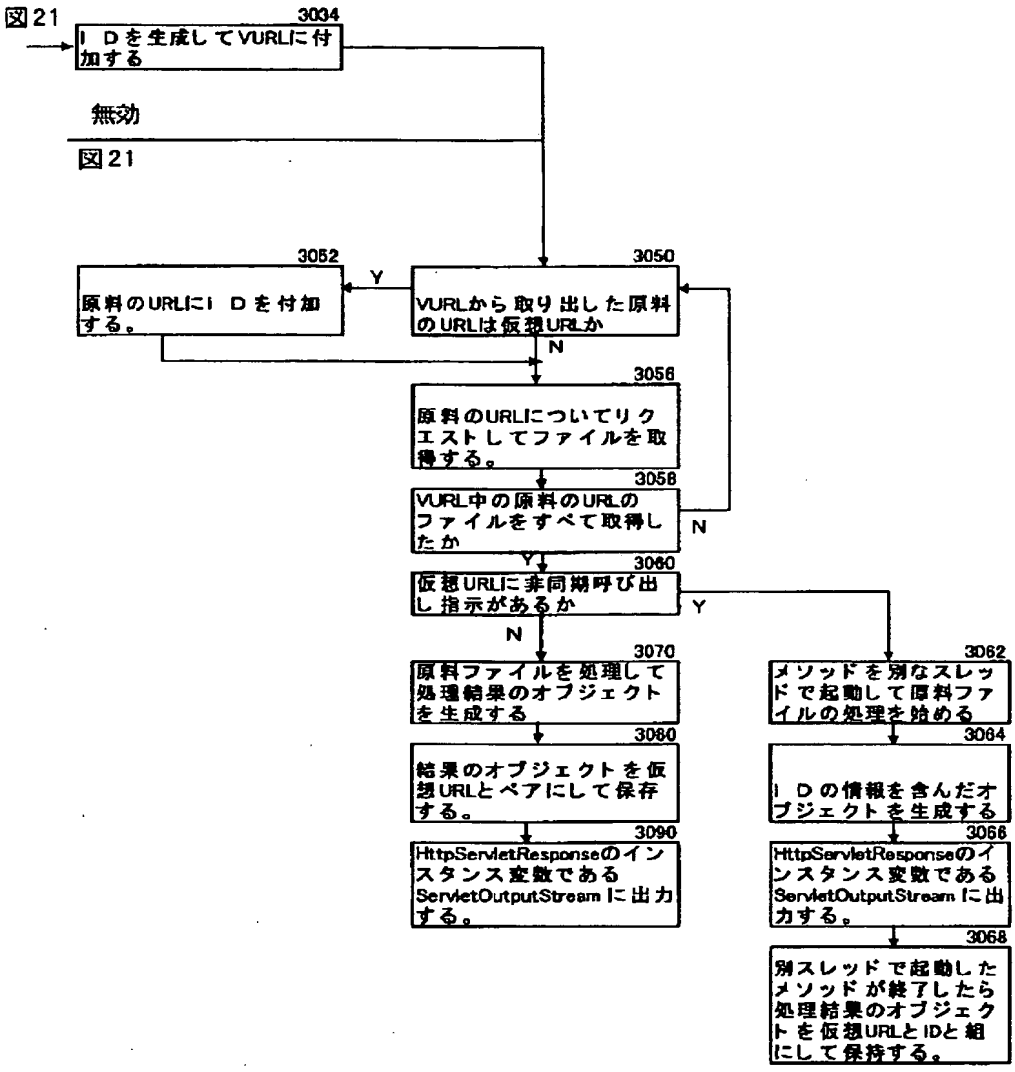
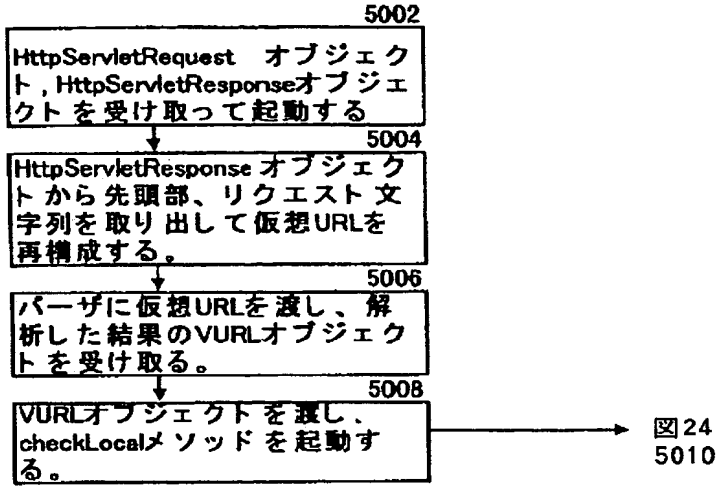
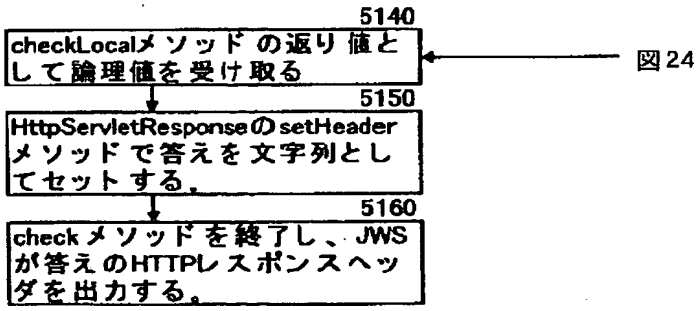


図21
3034
ＩＤを生成してVURLに付加する
無効
図21
3052
原料のURLにＩＤを付加する。
Y
3050
VURLから取り出した原料のURLは仮想URLか
N
3056
原料のURLについてリクエストしてファイルを取得する。
3058
VURL中の原料のURLのファイルをすべて取得したか
N
Y
3060
仮想URLに非同期呼び出し指示があるか
Y
N
3070
原料ファイルを処理して処理結果のオブジェクトを生成する
3080
結果のオブジェクトを仮想URLとペアにして保存する。
3090
HttpServletResponseのインスタンス変数であるServletOutputStreamに出力する。
3062
メソッドを別なスレッドで起動して原料ファイルの処理を始める
3064
ＩＤの情報を含んだオブジェクトを生成する
3066
HttpServletResponseのインスタンス変数であるServletOutputStreamに出力する。
3068
別スレッドで起動したメソッドが終了したら処理結果のオブジェクトを仮想URLとIDと組にして保持する。

【図２３】


5002
HttpServletRequest オブジェクト，HttpServletResponseオブジェクトを受け取って起動する
5004
HttpServletResponse オブジェクトから先頭部、リクエスト文字列を取り出して仮想URLを再構成する。
5006
パーザに仮想URLを渡し、解析した結果のVURLオブジェクトを受け取る。
5008
VURLオブジェクトを渡し、checkLocalメソッドを起動する。
図24
5010


5140
checkLocalメソッドの返り値として論理値を受け取る
図24
5150
HttpServletResponseのsetHeaderメソッドで答えを文字列としてセットする。
5160
checkメソッドを終了し、JWSが答えのHTTPレスポンスヘッダを出力する。

【図２４】

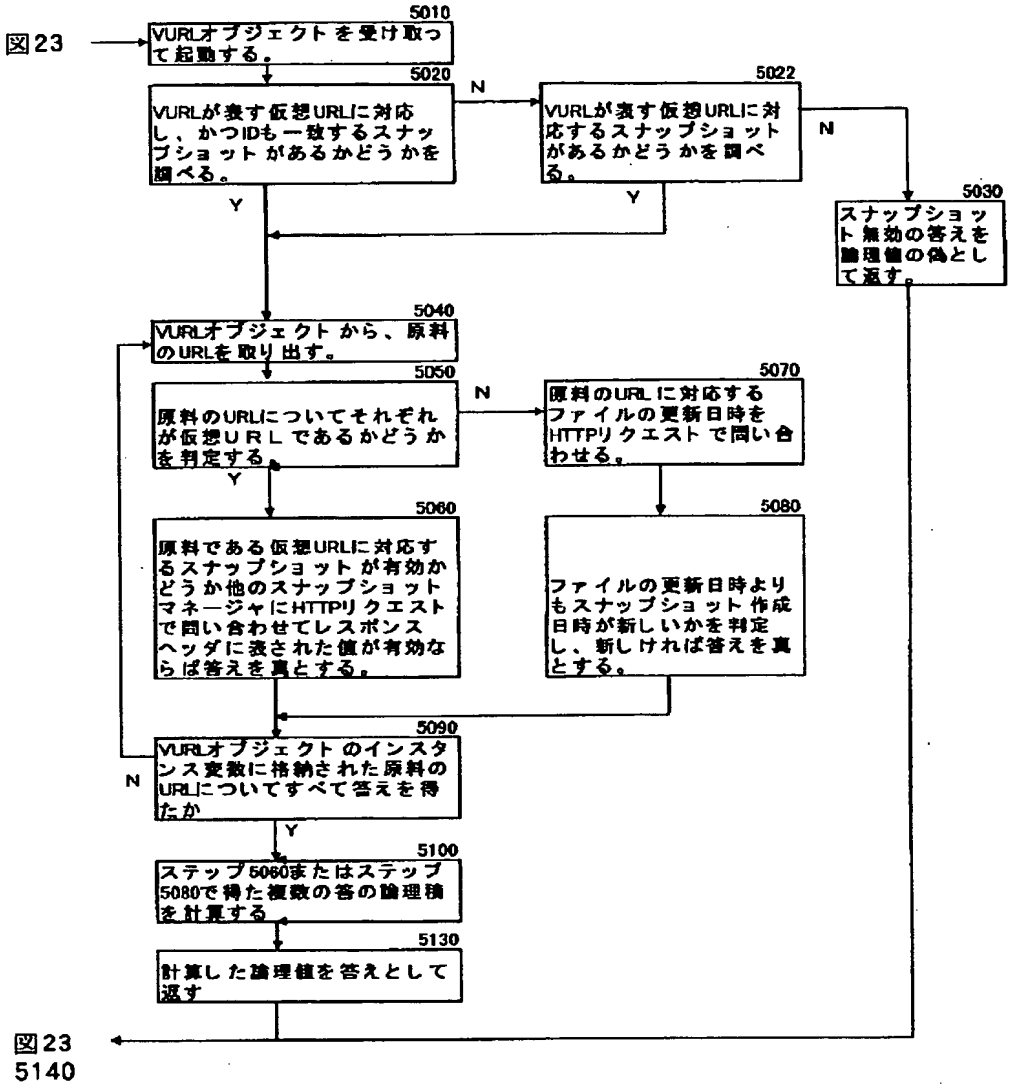


図23
5010
VURLオブジェクトを受け取って起動する。
5020
VURLが表す仮想URLに対応し、かつIDも一致するスナップショットがあるかどうかを調べる。
N
5022
VURLが表す仮想URLに対応するスナップショットがあるかどうかを調べる。
N
5030
スナップショット無効の答えを論理値の偽として返す。
Y
Y
5040
VURLオブジェクトから、原料のURLを取り出す。
5050
原料のURLについてそれぞれが仮想ＵＲＬであるかどうかを判定する。
N
5070
原料のURLに対応するファイルの更新日時をHTTPリクエストで問い合わせる。
Y
5060
原料である仮想URLに対応するスナップショットが有効かどうか他のスナップショットマネージャにHTTPリクエストで問い合わせてレスポンスヘッダに表された値が有効ならば答えを真とする。
5080
ファイルの更新日時よりもスナップショット作成日時が新しいかを判定し、新しければ答えを真とする。
5090
VURLオブジェクトのインスタンス変数に格納された原料のURLについてすべて答えを得たか
N
Y
5100
ステップ5060またはステップ5080で得た複数の答の論理積を計算する
5130
計算した論理値を答えとして返す
図23
5140